IJREAT International Journal of Research in Engineering & Advanced Technology, Volume 2, Issue 2, Apr-May, 2014
ISSN: 2320 – 8791 (Impact Factor: 1.479)
www.ijreat.org

Secure Routing Protocol in Sensor Networks for Vampire Attacks

Ms.T.Sumaiya Begum[1], Ms.S.Swathilakshmi[2], Mrs.S.Blessy[3]

[1, 2, 3]**Department of Information Technology, Anand Institute of Higher Technology, Chennai, Tamil Nadu, India**

Abstract

Ad hoc wireless sensor networks continually proves to exist as an exciting domain in carrying out various research activities and also has varied applications worldwide. However, the more promising it is, the more threats it carries with respect to its security. One such threat is a new class of resource depletion attacks often called as vampire attacks which drain the node's battery power and disable the network permanently. Much spotlight is given on this class of attacks as we will discuss two variations of vampire attacks along with a proof of concept protocol put forth by Parno, Luk, Gustad, Perrig (PLGP) which bounds the damage caused by vampire during the packet forwarding phase. But PLGP fails to offer a satisfactory solution during the network topology discovery phase. In this paper, we discuss methods to secure the discovery phase and prevent the occurrence of vampire attacks through an enhanced DSDV protocol.

1. Introduction

Wireless sensor networks promises new and diverse applications in the future which includes fields like environmental monitoring, agricultural monitoring, machine health monitoring, surveillance and medical monitoring. These networks, which connect the physical world with the digital world, provide us with a richer understanding of our environment and with the ability to more accurately control our surroundings. However, there are many challenges that must be addressed before the full potential of these networks are realized. Wireless sensor networks must be reliable and scalable to support large numbers of unattended wireless sensors; they must last for extended periods of time using limited battery power; they must be secure against outside attacks on the network and on data fidelity; they must be accurate in providing required information while performing in-network processing to reduce data load; and they must interface with existing networks. While these schemes can prevent attacks on the short-term availability of a network, they do not address attacks that affect long-term availability—the most permanent denial of service attack is to entirely deplete nodes' batteries. This is an instance of a resource depletion attack, with battery power as the resource of interest. In this paper, we consider how routing protocols, even those designed to be secure, lack protection from these attacks, which we call Vampire attacks, since they drain the life from networks nodes. These attacks are distinct from DoS, reduction of quality (RoQ), and routing infrastructure attacks as they do not disrupt immediate availability, but rather work over time to entirely disable a network.

2. Vampire Attacks

We define a Vampire attack as the composition and transmission of a message that causes more energy to be consumed by the network than if an honest node transmitted a message of identical size to the same destination, although using different packet headers. We measure the strength of the attack by the ratio of network energy used in the benign case to the energy used in the malicious case, i.e., the ratio of network-wide power utilization with malicious nodes present to energy usage with only honest nodes when the number and size of packets sent remains constant. Safety from Vampire attacks implies that this ratio is 1. Energy use by malicious nodes is not considered, since they can always unilaterally drain their own batteries.and moreover they are not protocol-specific, in that they do not rely on design properties or implementation faults of particular routing protocols, but rather exploit general properties of protocol classes such as link-state, distance-vector, source routing, and geographic and beacon routing. Neither do these attacks rely on flooding the network with large amounts of data, but rather try to transmit as little data as possible to achieve the largest energy drain, preventing a rate limiting solution. Since Vampires use protocol-compliant messages, these attacks are very difficult to detect and prevent.

3. Overview

In this paper, we present two variations of increasingly damaging Vampire attacks, one called as the carousel attack, where there are purposely intended loops and the packets move in circles without reaching the intended destination node. It targets source routing protocols by exploiting the limited verification of message headers at forwarding nodes,

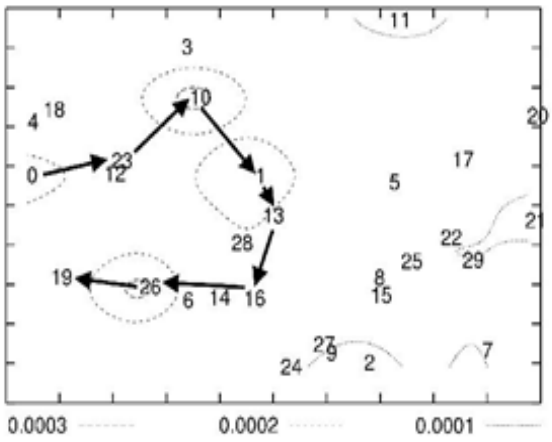
allowing a single packet to repeatedly traverse the same set of nodes. Much has been discussed about this attack but no countermeasures were provided. The second one called the stretch attack, where the path chosen for the packet to traverse is usually longer than the optimal path possibly traversing the entire network. It also targets source routing protocol and increases packet path lengths, causing packets to be processed by a number of nodes that is independent of hop count along the shortest path between the adversary and packet destination. A single attacker can use a carousel attack to increase energy consumption by as much as a factor of 4, while stretch attacks depends on the position of malicious nodes. When both these attacks are combined, the number of adversarial nodes in the network increases, thereby causing a big impact on the network. While carousel attack is simple to prevent with negligible over-head, the stretch attack is far more challenging. We evaluate their vulnerability, their aftermaths and suggestions for tackling them. We next discuss the PLGP protocol taken from the model by Eugene and Nicholas. This protocol bounds the damage caused during the packet forwarding phase. We then show how an adversary can target not only packet forwarding phase but also route and topology discovery phases—if discovery messages are flooded, an adversary can, for the cost of a single packet, consume energy at every node in the network which has proved to be a drawback in the case of PLGP protocol. In the later part of this paper, we discuss the DSDV protocol which is table driven and how it can be enhanced to prevent the damage which can arise during the topology discovery phase.

4. Attacks on Source Routing Protocol

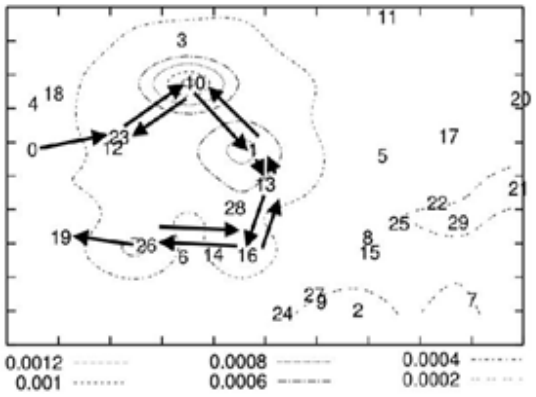
In this section, the carousel and stretch attacks are evaluated using a randomly generated network topology consisting of 30 nodes and a single randomly generated DSR agent, using the ns-2 network simulator.

4.1 Carousel Attack

In this attack, an adversary sends a packet with a route composed as a series of loops, such that the same node appears in the route many times which appears to be cyclic. This strategy can be used to increase the route length beyond the number of nodes in the network, only limited by the number of allowed entries in the source route. The same can be explained with the help of a diagram.



(a) Honest scenario: node 0 sends a single message to node 19.



(b) Carousel attack (malicious node 0): the nodes traversed by the packet are the same as in (a), but the loop over all forwarding nodes roughly triples the route length (the packet traverses the loop more than once). Note the drastically increased energy consumption among the forwarding nodes.

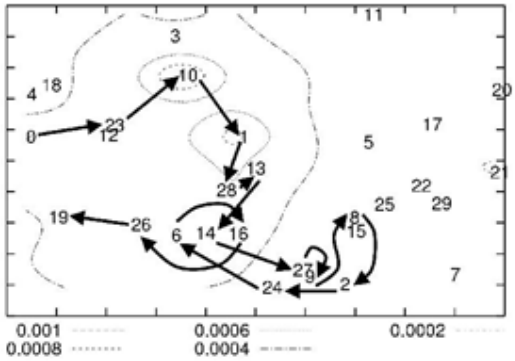
In the first diagram (a), the source node 0 transmits the message to the destination node 19 using the shortest optimal path in the topology. This is called an honest scenario. In the second diagram, the node 0 transmits the message to the destination node 19 but this time the source node becomes malicious and generates loops along the traversal path causing a tremendous increase in the energy of the network. There are many circular loops formed at node 10,1,16 and 26.

Assuming the adversary limits the transmission rate to avoid saturating the network, the theoretical limit of this attack is an energy usage increase factor of O(n) where n is the maximum route length. Overall energy consumption increases by up to a factor of 3.96 per message. On an average, a randomly located carousel attacker in the example topology can increase network energy consumption by a factor of 1:48 ± 0:99. The

reason for this large standard deviation is that the attack does not always increase energy usage—the length of the adversarial path is a multiple of the honest path, which is in turn, affected by the position of the adversary in relation to the destination, so the adversary's position is important to the success of this attack.

4.2 Stretch Attack

In this type of attack, the malicious node constructs paths to transmit the message to the destination node (which may be an honest one) which are far longer than the optimal path in the topology. These artificial routes can cause nodes that do not lie along the honest route to consume energy by forwarding packets they would not receive in honest scenarios. Due to this, they get to lose their own energy. This can be explained with the help of a diagram.

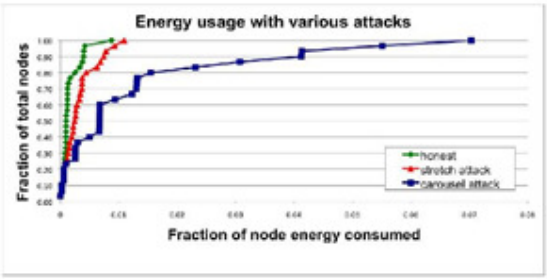


(c) Stretch attack (malicious node 0): the route diverts from the optimal path between source and destination, roughly doubling in length. Note that while the per-node energy consumption increase is not as drastic as in (b), the region of increased energy consumption is larger. Overall energy consumption is greater than in the carousel attack, but spread more evenly over more network nodes.

Comparing the diagram(c) with (a), the latter one is of an honest scenario where node 0 transmits a message to node 19 using the best optimal path available in the network topology. However in the former diagram(c), the malicious node constructs artificial long paths starting from the node 28 where a diversion takes place to nodes 13, 14, 27, 9, 8, 15, 2, 24 and 6 no longer using the optimal path.

4.3 Comparison

On comparing the usage of energy in the network by various attacks we observe the following,



The energy usage of carousel attacks is higher compared to other attacks because in carousel attack, the nodes along the shorter path only are affected. In contrast, the energy consumption due to stretch attack is somewhat uniform in the network because the traversal route is increased making most of the nodes in the topology spend their energy and process the packet.

The resource utilization was independently computed for honest nodes and malicious nodes and was found that the malicious nodes did not use a disproportionate amount of energy in carrying out the attack. In other words, malicious nodes are not driving down the cumulative energy of the network purely by their own use of energy. The theoretical limit of the stretch attack is a packet that traverses every network node, causing an energy usage increase of factor $O(\min(N, \lambda))$, where N is the number of nodes in the network and λ is the maximum path length allowed. This attack is potentially less damaging per packet than the carousel attack, as the number of hops per packet is bounded by the number of network nodes. However, adversaries can combine carousel and stretch attacks resulting in a "stretched cycle" where the packet travels in long paths and takes circular loops as well. Therefore, even if stretch attack protection is not used, route loops should still be detected and removed to prevent the combined attack. Not all routes can be significantly lengthened, depending on the location of the adversary. Unlike the carousel attack, where the relative positions of the source and sink are important, the stretch attack can achieve the same effectiveness independent of the attacker's network position relative to the destination, so the worst case effect is far more likely to occur.

Both these attacks are less effective in hierarchical networks due to the presence of aggregators. In hierarchical networks, the nodes send messages to aggregators, who in turn send them to other aggregators, and they route it to the monitoring point. The described carousel and stretch attacks are valid only within the network neighbourhood of the adversarial node. If an adversary corrupts nodes intelligently or controls a small but non-trivial portion of nodes, it can execute these attacks within the individual network neighbourhoods. A single adversary per neighbourhood is enough to disable the entire network.

4.4 Countermeasures

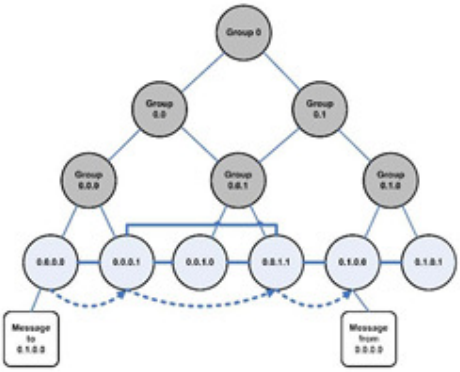
The carousel attack can be prevented when the nodes that are responsible for forwarding the message are enabled with mechanism to check the source routes for loop. But it comes with the cost of extra forwarding logic and more overhead, yet it is worthwhile in malicious environments. The ns-2 DSR protocol implements loop detection, but confusingly does not use it to check routes in forwarded packets. When a loop is detected in the network, the source route is corrected and the packet is sent on, but one of the attractive features of source routing is that the route can itself be signed by the source. Therefore, it is better to simply drop the packet, especially considering that the sending node is likely malicious (honest nodes should not introduce loops). An alternate solution provided is to alter the processing of message by the intermediate nodes. To forward a message, a node must determine the next hop by locating itself in the source route. If a node searches for itself from the destination backward instead from the source forward, any loop that includes the current node will be automatically truncated (the last instance of the local node will be found in the source route rather than the first). No extra processing is required for this since a node must perform this check anyway.

The stretch attack is far more challenging to prevent since the forwarding nodes do not check for optimality of the route. If this is called strict source routing because of no optimization and the route is followed exactly as specified in the header, then loose source routing can be defined. In loose source routing, the intermediate nodes can replace the entire route or some part of the route if they know of any other better traversal path to the intended destination. To accomplish this, the nodes should discover and cache optimal routes to at least some fraction of other nodes, partially defeating the as-needed discovery advantage. Moreover, caching must be done carefully lest a maliciously suboptimal route be introduced. Loose source routing was simulated in a randomly generated topology of up to 1,000,000 nodes, with diameter of 10-14. Results showed that the amount of node-local storage required for achieving reasonable levels of mitigation approaches global topology knowledge, defeating the purpose of using source routing. The number of nodes traversed by loose source-routed packets is suboptimal by at least a factor of 10, with some routes approaching a factor of 50. Only a few messages encountered a node with a better path to the destination than the originally assigned long source route. Therefore it was concluded that loose source routing is worse than keeping global state at every node.

Alternatively, the damage of carousel and stretch attackers can be bounded by limiting the allowed source route length based on the expected maximum path length in the network, a way to determine the network diameter is needed. While there are suitable algorithms there has been very little work on whether they could yield accurate results in the presence of adversaries. If the number of nodes is known ahead of time, graph-theoretic techniques can be used to estimate the diameter. Rate limiting may initially seem to be good but later was found that it was not ideal. It limits malicious sending rate, potentially increasing network lifetime, but that increase becomes the maximum expected lifetime, since adversaries will transmit at the maximum allowed rate. Moreover, sending rate is already limited by the size of nodes' receive queues in rate-unlimited networks. Rate limiting also potentially punishes honest nodes that may transmit large amounts of time-critical data, but will send little data over the network lifetime.

5 Clean State Sensor Network Routing

A clear slate sensor network routing protocol proposed by Parno, Luk, Gustad and Perrig(called PLGP), is modified to resist Vampire attacks during the packet forwarding phase in this section. It was originally developed for security but is vulnerable to Vampire attacks. This PLGP protocol consists of two phases a topology discovery phase, followed by a packet forwarding phase, with the former optionally repeated on a fixed schedule to ensure that topology information stays current. This protocol is explained with the help of a diagram,



5.1 Topology Discovery Phase

PLGP begins with the discovery of the network topology where a tree is formed which is used as an addressing scheme. When discovery begins, each node has a limited view of the network—the node knows only itself. It is a time-limited period during which every node must announce its presence by broadcasting a certificate of identity, including its public key (referred to as node ID), signed by a trusted offline authority. Each node starts as its own group of size one, with a virtual address 0. Nodes who overhear presence broadcasts form groups with their neighbours. When two individual nodes (each with an initial address 0) form a group of size two, one of them takes the address 0, and the other becomes 1. Groups merge preferentially with the smallest neighbouring group,

which may be a single node. Like individual nodes, each group will initially choose a group address 0, and will choose 0 or 1 when merging with another group. Each group member prepends the group address to their own address, e.g., node 0 in group 0 becomes 0.0, node 0 in group 1 becomes 1.0, and so on. Each time two groups merge, the address of each node is lengthened by 1 bit. Implicitly, this forms a binary tree of all addresses in the network, with node addresses as leaves. At the end of discovery, each node learns every other node's virtual address, public key, and certificate, since every group members knows the identities of all other group members and the network converges to a single group. Each node computes the same address tree as other nodes. All leaf nodes in the tree are physical nodes in the network, and their virtual addresses correspond to their position in the tree (as given in the diagram). All nodes learn each other's virtual addresses and cryptographic keys. The final address tree is verifiable after network convergence, and all forwarding decisions can be independently verified. Furthermore, assuming each legitimate network node has a unique certificate of membership (assigned before network deployment), nodes who attempt to join multiple groups, produce clones of themselves in multiple locations, or otherwise cheat during discovery can be identified and evicted.

5.4 Packet Forwarding Phase

During the packet forwarding phase, all decisions are made independently by each node. When receiving a packet, a node determines the next hop by finding the most significant bit of its address that differs from the message originator's address (shown in the diagram). Thus, every forwarding event (except when a packet is moving within a group in order to reach a gateway node to proceed to the next group) shortens the logical distance to the destination, since node addresses should be strictly closer to the destination.

5.5 PLGP Protocol with Vampire Attack

In PLGP, the forwarding nodes do not know what path a packet took. This allows adversaries to divert packets to any part of the network, even if that area is logically further away from the destination than the malicious node. This makes PLGP vulnerable to Vampire attacks. Considering the directional antenna attack, a receiving honest node may be farther away from the packet destination than the malicious forwarding node, but the honest node has no way to tell that the packet it just received is moving away from the destination; the only information available to the honest node is its own address and the packet destination address, but not the address of the previous hop (who can lie). Thus, the Vampire can move a packet away from its destination without being detected. The situation is worse if the packet returns to the Vampire in the process of being forwarded—it can now be rerouted again, causing something similar to the carousel attack. Recalling that the damage from the carousel attack is bounded by the maximum length of the source route, but in PLGP the adversary faces no such limitation, so the packet can cycle indefinitely. Nodes may sacrifice some local storage to retain a record of recent packets to prevent this attack from being carried out repeatedly with the same packet. Random direction vectors, as suggested in PLGP, would likewise alleviate the problem of indefinite cycles by avoiding the same malicious node during the subsequent forwarding round.

5.6 Modified PLGP

After having found that the original version of PLGP is vulnerable to vampire attacks, it was modified into PLGPa (PLGP with attestations) to resist vampire attacks during the packet forwarding phase. A property called the "no backtracking" property is introduced in this context. This property is satisfied for a given packet if and only if it consistently makes progress toward its destination in the logical network address space. No-backtracking implies Vampire resistance. The packet progress is checked independently where the nodes keep a record of route cost and communicate the local cost to the next hop. The next hop verifies if the remaining route cost is less than before and if the route cost is less then, the packet is progressing towards the destination and there is no malicious attack. This allows us to bound the attack in the network.

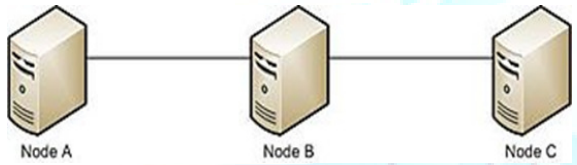
PLGP does not satisfy no-backtracking. PLGP differs from other protocols in that packets paths are further bounded by a tree, forwarding packets along the shortest route through the tree that is allowed by the physical topology. In other words, packet paths are constrained both by physical neighbour relationships and the routing tree. Since the tree implicitly mirrors the topology and every node holds an identical copy of the address tree, every node can verify the optimal next logical hop. However, this is not sufficient for no-backtracking to hold, since nodes cannot be certain of the path previously traversed by a packet. Communicating a local view of route cost is not as easy as it seems, since adversaries can always lie about their local metric, and so PLGP is still vulnerable to attacks, which allow adversaries to divert packets to any part of the network.

This PLGP was modified a little to preserve the property of no-backtracking. A verifiable path history was added to every PLGP packet. This resulted in PLGPa (PLGP with attestations). PLGPa uses this packet history together with PLGP's tree routing structure so every node can securely verify progress, preventing any significant adversarial influence on the path taken by any packet which traverses at least one honest node. Whenever node n forwards packet p, it

does this by attaching a nonreplayable attestation (signature). These signatures form a chain attached to every packet, allowing any node receiving it to validate its path. Every forwarding node verifies the attestation chain to ensure that the packet has never traveled away from its destination in the logical address space. PLGPa satisfies no-backtracking.

6. DSDV Protocol

Destination-Sequenced Distance-Vector Routing (DSDV) is a table-driven routing scheme for ad hoc mobile networks based on the Bellman–Ford algorithm. It was developed by C. Perkins and P. Bhagwat in 1994. The main contribution of the algorithm was to solve the routing loop problem.



The above diagram shows a simple network with three nodes. Every mobile station maintains a routing table and each entry in the routing table contains all the available destinations, number of hops to reach the destination and sequence number assigned by the destination node. The sequence number is generally even if a link is present else it is odd and this sequence number is used to distinguish stale routes from new ones avoiding the formation of loops. The number is generated by the destination, and the emitter needs to send out the next update with this number.

The stations periodically transmit their routing tables to their immediate neighbours. A station also transmits its routing table if a significant change has occurred in its table from the last update sent. So, the update is both time-driven and event-driven. The routing table updates can be sent in two ways. One way is a "full dump" and the other is an incremental update. A full dump sends the full routing table to the neighbours and could span many packets whereas in an incremental update only those entries from the routing table are sent that has a metric change since the last update and it must fit in a packet. If there is space in the incremental update packet then those entries may be included whose sequence number has changed. When the network is relatively stable, incremental updates are sent to avoid extra traffic and full dump are relatively infrequent. In a fast-changing network, incremental packets can grow big so full dumps will be more frequent.

Each route update packet, in addition to the routing table information, also contains a unique sequence number assigned by the transmitter. The route with the highest (i.e. most recent) sequence number is used. If two routes have the same sequence number then the route with the best metric (i.e. shortest route) is used. Based on the past history, the stations estimate the settling time of routes. The stations delay the transmission of a routing update by settling time so as to eliminate those updates that would occur if a better route were found very soon. The only flaw is that DSDV requires a regular update of its routing tables, which uses up battery power and a small amount of bandwidth even when the network is idle and whenever the topology of the network changes, a new sequence number is necessary before the network re-converges.

7. Enhanced DSDV protocol for Topology Discovery

PLGPa did bound vampire attack during the packet forwarding phase but the fact that it includes path attestations increases the packet size ,incurs penalties in bandwidth use and radio power too increases. And adding extra verification in the intermediate nodes also increases processor utilization, time and power. All this worthwhile when an adversary is present in the network else even an entirely normal network may run at a relatively low speed. While PLGPa is not vulnerable to Vampire attacks during the forwarding phase, it is not the same with the topology discovery phase and the bound to be placed during the discovery phase is still unknown. And moreover PLGPa can only resist the presence of adversaries when detected. They cannot prevent its occurrence completely.

In this paper, inputs from DSDV are taken and modified to make it an Enhanced DSDV protocol to provably secure the network topology discovery phase. The table-driven protocol is used in this context to provide secure topology discovery of the network. We discuss the three basic features of DSDV protocol. Since DSDV is table-driven, it requires regular update of its routing tables and this in turn uses the battery power and extra bandwidth even when the network is in an idle state. Next feature is that it is pro-active. DSDV doesn't just respond but rather it controls the entire network. This may lead to slow convergence and slow propagation. And the third feature is that it is purely "anytime active" protocol. The last feature since its active all the time, it can use battery power and bandwidth for its own sake apart from the networks sake.

In an attempt to make the network more secure in the discovery phase, we have proposed modifications in DSDV protocol and implemented it using the ns-2 simulator to show the results. We have replaced the table system of DSDV with an index system which will not require frequent and regular

updating of the status of the network. The updating process will take place only as and when required. This will help in cutting down the cost of battery and bandwidth usage. This makes DSDV from being pro-active to becoming on-demand. It does not have to manage the entire network as there will not be any table with regular updates. And lastly the protocol does not have to stay active all time and can be called by the sensor nodes only when there is a necessity for a connection establishment between the nodes. This makes it simple on the network which can be prevented from wastage of energy and bandwidth.

8. Conclusion

In this paper, we have presented the stretch and carousel attacks with a model of randomly generated network topology and then we have briefed about the PLGP protocol by Parno, Luk, Gustad and Perrig, all of these defined by Eugene and Nicholas. This protocol bounds the damage caused during the packet forwarding phase. We then showed how an adversary can target not only packet forwarding phase but also route and topology discovery phases—if discovery messages are flooded, an adversary can, for the cost of a single packet, consume energy at every node in the network which has proved to be a drawback in the case of PLGP protocol. We have then come up with an enhanced DSDV protocol, which can possibly avoid the occurrence of vampire attacks during the discovery of the topology phase. We have showed the modifications done to the DSDV protocol via ns-2 simulator and have prevented vampire attacks from making its presence during the network topology discovery phase.

References

[1] Eugene Y.Vasserman and Nicholas Hopper, Vampire Attacks: Draining Life from Wireless Ad Hoc Sensor Networks.
[2] Wikipedia.
[3] Destination sequenced distance vector (DSDV) protocol. Guoyou He, Networking Laboratory, Helsinki University of Technology, ghe@cc.hut.fi
[4] Routing Protocols for Ad Hoc Mobile Wireless Networks, PadminiMisra, misra@cse.wustl.edu