CHAPTER II

LITERATURE SURVEY

2.0 INTRODUCTION

Applications of security in MANET range from military tactical operations to civil rapid development such as data collection / sensor networks and instantaneous classroom / official meeting room applications. Intrusion detection is to detect malicious activities that compromise the security of a computer network system [1]. There exist mainly two categories of intrusion detection techniques: anomaly detection and signature recognition (misuse detection) [65]. Anomaly detection techniques detect an intrusion when the observed activities demonstrate a large deviation from the profile of normal activities. Signature recognition techniques store patterns of intrusion signatures and compare those signatures with the observed activities for a match to detect an intrusion.

2.1 INTRUSION DETECTION AND ATTACKS

ID (Intrusion Detection) in MANET can be forseen as a problem of illicit node pattern behavioral classification, while the system should also deal with few intrinsic characteristics that make it very difficult to detect intrusions directly using classical pattern recognition methods. An example shows where normal and anomalous states are distinguished using features that are multi-dimensional and extreme asymmetric nature is identified in the amount of data available for
these two sets of states. Furthermore, the patterns involved cannot be recognized by linear methods. Hence the need to develop new architecture and mechanisms to protect the wireless networks and mobile computing applications is accepted. The malicious node(s) can attack MANET using different ways, such as sending fake messages several times, fake routing information, and advertising fake links to disrupt routing operations. In the following subsection, current routing attacks and its countermeasures against MANET protocols are discussed in detail.

2.1.1 Flooding Attack [71], [80]

In flooding attack, attacker exhausts the network resources, such as bandwidth and to consume a node’s resources, such as computational and battery power or to disrupt the routing operation to cause severe degradation in network performance. For example, in AODV protocol, a malicious node can send a large number of RREQs in a short period to a destination node that does not exist in the network. Since, no one will reply to the RREQs, these RREQs will flood the whole network. As a result, all of the node battery power, as well as network bandwidth will be consumed and could lead to DoS.

A simple mechanism is proposed to prevent the flooding attack in the AODV protocol [25]. In this approach, each node monitors and calculates the rate of its neighbors’ RREQ. If the RREQ rate of any neighbor exceeds the predefined threshold, the node records the ID of
this neighbor in a blacklist. Then, the node drops any future RREQs from nodes that are listed in the blacklist. The limitation of this approach is that it cannot prevent against the flooding attack in which the flooding rate is below the threshold. Another drawback of this approach is that if a malicious node impersonates the ID of a legitimate node and broadcasts a large number of RREQs, other nodes might put the ID of this legitimate node on the blacklist by mistake. In [5], the authors show that a flooding attack can decrease throughput by 84 percent. The authors proposed an adaptive technique to mitigate the effect of a flooding attack in the AODV protocol. This technique is based on statistical analysis to detect malicious RREQ floods and avoid the forwarding of such packets.

Similar to [25], in this approach, each node monitors the RREQ it receives and maintains a count of RREQs received from each sender during the preset time period. The RREQs from a sender whose RREQ rate is above the threshold will be dropped without forwarding. Unlike the method proposed in [27], where the threshold is set to be fixed, this approach determines the threshold based on a statistical analysis of RREQs. The key advantage of this approach is that it can reduce the impact of the attack for varying flooding rates.

2.1.2 Blackhole Attack [72], [88]

In blackhole attack, a malicious node sends fake routing information, claiming that it has an optimum route and causes other
good nodes to route data packets through the malicious one. For example, in AODV, the attacker can send a fake RREP (including a fake destination sequence number that is fabricated to be equal or higher than the one contained in the RREQ) to the source node, claiming that it has a sufficiently fresh route to the destination node. This causes the source node to select the route that passes through the attacker. Therefore, all traffic will be routed through the attacker, and therefore, the attacker can misuse or discard the traffic. Fig. 2.1 shows an example of a blackhole attack, where attacker A sends a fake RREQ to the source node S, claiming that it has a sufficiently fresher route than other nodes. Since the attacker’s advertised sequence number is higher than other nodes’ sequence numbers, the source node S will choose the route that passes through node A.

![Diagram showing blackhole attack in AODV](image)

**Fig. 2.1 Blackhole attack on AODV**
The route CREQ (Confirmation Request) and route CREP (Confirmation Reply) is introduced in [15] to avoid the blackhole attack. In this approach, the intermediate node not only sends RREPs to the source node but also sends CREQs to its next-hop node toward the destination node. After receiving a CREQ, the next-hop node looks up its cache for a route to the destination. If it has the route, it sends the CREP to the source. Upon receiving the CREP, the source node can confirm the validity of the path by comparing the path in RREP and the one in CREP. If both are matched, the source node judges that the route is correct. One drawback of this approach is that it cannot avoid the blackhole attack in which two consecutive nodes work in collusion, that is, when the next-hop node is a colluding attacker sending CREPs that support the incorrect path. In [41], the authors proposed a solution that requires a source node to wait until a RREP packet arrives from more than two nodes. Upon receiving multiple RREPs, the source node checks whether there is a shared hop or not. If there is shared hop, the source node judges that the route is safe. The main drawback of this solution is that it introduces time delay, because it must wait until multiple RREPs arrive. In another attempt [88], [94], the authors analyzed the blackhole attack and showed that a malicious node must increase the destination sequence number sufficiently to convince the source node that the route provided is sufficiently enough. Based on this analysis, the authors propose a statistical based anomaly detection approach to detect the blackhole
attack, based on differences between the destination sequence numbers of the received RREPs. The key advantage [56], [70], [99] of this approach is that it can detect the attack at low cost without introducing extra routing traffic, and it does not require modification of the existing protocol. However, false positives are the main drawback of this approach due to the nature of anomaly detection.

2.1.3 Link Spoofing Attack [9], [87], [90]

In a link spoofing attack, a malicious node advertises fake links with non-neighbors to disrupt routing operations. For example, in the OLSR (Optimized Link State Routing Protocol) protocol, an attacker can advertise a fake link with a target’s two-hop neighbors. This causes the target node to select the malicious node to be its MPR (Multi Point Relay). As an MPR node, a malicious node can then manipulate data or routing traffic, for example, modifying or dropping the routing traffic or performing other types of DoS attacks. Fig. 2.2 shows an example of the link spoofing attack in an OLSR MANET [20]. In the figure, we assume that node A is the attacking node, and node T is the target to be attacked. Before the attack, both nodes A and E are MPRs for node T. During the link spoofing attack, node A advertises a fake link with node T’s two-hop neighbor, that is, node D. According to the OLSR protocol, node T will select the malicious node A as its only MPR since node A is the minimum set that reaches node T’s two-hop neighbors. By being node T’s only MPR, node A can then drop or withhold the routing traffic generated by node T.
Fig. 2.2 Link spoofing attack

A location information-based detection method is proposed [22] to detect link spoofing attack by using cryptography with a GPS and a time stamp. This approach requires each node to advertise its position obtained by the GPS and the time stamp to enable each node to obtain the location information of the other nodes. This approach detects the link spoofing by calculating the distance between two nodes that claim to be neighbors and checking the likelihood that the link is based on a maximum transmission range. The main drawback of this approach is that it might not work in a situation where all MANET nodes are not equipped with a GPS. Furthermore, attackers can still advertise false information and make it hard for other nodes to detect the attack. In [8], the authors show that a malicious node that advertises fake links with a target’s two-hop neighbors can successfully make the target choose it as the only MPR. Through simulations, the authors show that link spoofing can have a devastating impact on the target node. Then, the authors present a technique to detect the link spoofing attack by adding two-hop information to a HELLO message. In
particular, the proposed solution requires each node to advertise its
two-hop neighbors to enable each node to learn complete topology up
to three hops and detect the inconsistency when the link spoofing
attack is launched. The main advantage of this approach is that it can
detect the link spoofing attack without using special hardware such as
a GPS or requiring time synchronization. One limitation of this
approach is that it might not detect link spoofing with nodes further
away than three hops.

2.1.4 Wormhole Attack

A wormhole attack [13] is one of the most sophisticated and
severe attacks in MANETs. In this attack, a pair of colluding attackers
record packets at one location and replay them at another location
using a private high speed network. The seriousness of this attack is
that it can be launched against all communications that provide
authenticity and confidentiality. In wormhole attack, we assume that
nodes A1 and A2 are two colluding attackers and that node S is the
target to be attacked. During the attack, when source node S
broadcasts an RREQ to find a route to a destination node D, its
neighbors C and E forward the RREQ as usual. However, node A1,
which received the RREQ, forwarded by node C, records and tunnels
the RREQ to its colluding partner A2. Then, node A2 rebroadcasts this
RREQ to its neighbor H. Since this RREQ passed through a high speed
channel, this RREQ will reach node D first. Therefore, node D will
choose route D-H-C-S to unicast an RREP to the source node S and
ignore the same RREQ that arrived later. As a result, S will select route S-H-D that indeed passed through A1 and A2 to send its data.

In [13], packet leashes are proposed to detect and defend against the wormhole attack. In particular, the authors proposed two types of leashes: temporal leashes and geographical leashes. For the temporal leash approach, each node computes the packet expiration time $te$, based on the speed of light $c$ and includes $te$, in its packet to prevent the packet from traveling further than a specific distance, $L$. The receiver of the packet checks whether or not the packet expires by comparing its current time and the $te$ in the packet. The authors also proposed TIK, which is used to authenticate the expiration time that can otherwise be modified by the malicious node. The main drawback of the temporal leash is that it requires all nodes to have tightly synchronized clocks. For the geographical leash, each node must know its own position and have loosely synchronized clocks. In this approach, a sender of a packet includes its current position and the sending time. Therefore, a receiver can judge neighbor relations by computing distance between itself and the sender of the packet. The advantage of geographic leashes over temporal leashes is that the time synchronization need not be highly tight.

In [22], [52], [103] the authors offer protection against a wormhole attack in the OLSR protocol. This approach is based on location information and requires the deployment of a public key
infrastructure and time-stamp synchronization between all nodes that is similar to the geographic leashes proposed in [13]. In this approach, a sender of a HELLO message includes its current position and current time in its HELLO message. Upon receiving a HELLO message from a neighbor, a node calculates the distance between itself and its neighbor, based on a position provided in the HELLO message. If the distance is more than the maximum transmission range, the node judges that the HELLO message is highly suspicious and might be tunneled by a wormhole attack. In [21], the authors propose a SAM (Statistical Analysis of Multipath) which is an approach to detect the wormhole attack by using multipath routing. This approach determines the attack by calculating the relative frequency of each link that appears in all of the obtained routes from one route discovery. In this solution, a link that has the highest relative frequency is identified as the wormhole link. The advantage of this approach is that it introduces limited overhead when applied in multipath routing. However, it might not work in a non-multipath routing protocol, such as a pure AODV protocol.

2.1.5 Colluding Misrelay Attack

In colluding misrelay attack, multiple attackers work in collusion to modify or drop routing packets to disrupt routing operation in a MANET. This attack is difficult to detect by using the conventional methods such as watchdog [37], [73] and pathrater [16]. Consider the case where node A1 forwards routing packets for node T.
The first attacker A1 forwards routing packets as usual to avoid being detected by node T. However, the second attacker A2 drops or modifies these routing packets. In [8], [96] the authors discuss this type of attack in OLSR protocol and show that a pair of malicious nodes can disrupt up to 100 percent of data packets in the OLSR MANET.

A conventional acknowledgment-based approach might detect this type of attack in a MANET, especially in a proactive MANET, but because routing packets destined to all nodes in the network require all nodes to return an ACK (Acknowledgement), this could lead to a large overhead, which is considered to be inefficient. In [9], the author proposes a method to detect an attack in which multiple malicious nodes attempt to drop packets by requiring each node to tune their transmission power when they forward packets. As an example, the author studies the case where two colluding attackers drop packets. The proposed solution requires each node to increase its transmission power twice to detect such an attack. However, this approach might not detect the attack in which three colluding attackers work in collusion. In general, the main drawback of this approach is that even if we require each node to increase transmission power to be K times, we still cannot detect the attack in which K + 1 attackers work in collusion to drop packets. Therefore, further work must be done to counter against this type of attack efficiently.
2.2 SECURITY BASED PROTOCOLS USED IN MANET

[i] ARIADNE [73]

It is a proposal by Hu, Perrig and Johnson, and is based on DSR. ARIADNE [38] relies only on efficient symmetric cryptography. It ensures the authentication and the integrity of the routing packets:

[a] The destination node of a route discovery process can authenticate the source node [75].

[b] The source node can authenticate each intermediate node present on the path to the destination in the RREP message, and can ensure that no intermediate node is removed from the node list in the RREQ or RREP messages.

ARIADNE can authenticate routing messages using one of three schemes: (i) shared keys between each pair of nodes, (ii) shared keys between communicating nodes combined with broadcast authentication, and (iii) digital signatures. ARIADNE’s authors assume that there exists a key distribution scheme for each authentication scheme. This thesis discuss the use of ARIADNE with TESLA (Timed Efficient Stream Loss-tolerant Authentication) [21], an efficient broadcast authentication scheme that requires time synchronization. Using pair-wise shared keys avoids the need for synchronization, but at the cost of higher key setup overhead; broadcast authentication such as TESLA also allows some additional protocol optimizations.
ARIADNE needs a mechanism to enable each node to share a secret key (i.e., ksd between source and destination). A TESLA key for each node in the network for each node must be securely set up for each node in the network.

[ii] TESLA broadcast authentication protocol

ARIADNE uses the TESLA [29] broadcast authentication protocol for authenticating routing packets. TESLA is efficient in a way that it adds only a MAC (Message Authentication Code) to a routing packet in order to achieve the broadcast authentication. A MAC can provide point-to-point authentication between two nodes using the same shared key [30],[40],[49],[87],[106]. However, for broadcast communication, the receiving nodes need to know the MAC key to authenticate the message. This is a vulnerability that may allow any receiving node to forge packets and impersonate the sender. TESLA solves this issue by relying on the clock synchronization and delayed key disclosure.

In order to use TESLA, each sending node generate a one-way key chain by choosing an initial TESLA key $K_N$ and repeatedly applying a one-way hash function $H$ on this initial value. The equation is $K_i = H[K_{i+1}] = H^{N-i}[K_N]$ [21]. To authenticate any received value on the one-way chain, a node applies this equation to the received value to check if the computed value matches a previous received key on the chain. For example, in order to authenticate $K_i$, we use the equation
K_j = H^j[K_i] to compute a value of K_j. If this value matches the previously received value of K_j, then K_i is authenticated.

Each sending node decides a schedule to disclose each key of its one-way key chain, in the order K_0, K_1, ..., K_N. A simple key disclosure schedule, for example, would be to publish key K_i at time T_0 + i*t, where T_0 is the time at which K_0 is disclosed, and t is the key disclosure interval. TESLA relies on a receiving node to check which keys, a sending node may have already disclosed. To do it, a receiving node calculates the time synchronization between nodes. For example, let D be the maximum difference between any two nodes; the value D must be known by all nodes. To send a packet, the sending node picks a key K_i from its one-way key chain, uses the key to generate a MAC value. This MAC value is attached to the packet. On receiving a packet authenticated with TESLA, the receiving node checks if the K_i has been disclosed by verifying

\( t_r <= (T_0 + i^*t - D) \) – which is called TESLA condition. If this inequality is true, that means K_i has not yet been disclosed. Otherwise the key may have already disclosed and an attacker may have forged the packet. However, if this check is successful, the receiving node buffers the packet and waits for the sender to publish key K_\text{i-}; when the receiver receives K_\text{i-}, it first authenticates K_i by using the equation \( K_j = H^{i-j}[K_i] \), and then authenticates stored packets authenticated with a key K_j, where j <= i.
**Route Discovery phase**

In ARIADNE, the basic RREQ message contains eight fields, that are used to provide authentication and integrity to the routing protocol.

<ROUTE REQUEST, initiator, target, id, time interval, hash chain, node list, MAC list>.

The initiator and target are the address of the source and the destination nodes respectively. Like DSR, the source sets the id to an identifier that it has not recently used in initiating a RREQ. The time interval is a TESLA related parameter that is the pessimistic expected arrival time of the request at the destination. The source of the RREQ then initializes the hash chain to \( \text{MAC}_{\text{ksd}} \) (initiator, target, id, time interval), the node list and the MAC list to empty lists.

When a node A receives a RREQ for which it is not the destination, the node checks its local table of <initiator, id> values from recent requests it has received, to see if it has already received the same RREQ. If it has, the node discards the packet, as in DSR. The node also checks whether the time interval in the request is valid, based on the following TESLA condition: the key corresponding to it must not have been disclosed yet. If the TESLA condition is not met, the node discards the packet. Otherwise, the node modifies the request by appending its own address (A) to the node list in the request, replacing the hash chain field with \( H[A, \text{hash chain}] \), and appending
a MAC of the entire RREQ to the MAC list. The node uses the TESLA key $K_{Ai}$ to compute the MAC, where $i$ is the index for the time interval specified in the request. Finally, the node rebroadcasts the modified RREQ.

When the destination node (aka. the target node) receives the RREQ, it checks the TESLA condition, and that the hash chain field is equal to:

$$H [h_n , H [h_{n-1} , H [ . . . , H [h_1 , MAC_{Ksd} (initiator, target, id, time interval) ] . . . ] ] ]$$

where $h_i$ is the node address at position $i$ of the node list in the RREQ, and $n$ is the number of nodes in the node list. If the target node determines that the RREQ is valid, it returns a RREP to the source node, containing eight fields:

<ROUTE REPLY, target, initiator, time interval, node list, MAC list, target MAC, key list>

---

<table>
<thead>
<tr>
<th>Route to be found: S -&gt; A -&gt; B -&gt; C -&gt; D</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>RREQ Message $M = (Request, S, D, id, ti)$</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Time</th>
<th>Actions</th>
</tr>
</thead>
</table>
| 1    | $S: h_0 = MAC_{Ksd} (M)$  
$S$ broadcasts: $(M, h_0, (), ())$ |
| 2    | $A: h_1 = H(A, h_0)$  
$M_A = MAC_{KAt} (M, h_0, (A), ())$  
$A$ broadcasts: $(M, h_1, (A), (M_A))$ |
### An example of ARIADNE Route Discovery process

The SAODV protocol [72] was proposed to answer the challenge of securing a MANET network. SAODV is an extension of the AODV routing protocol, and it can be used to protect the route discovery mechanism by providing security features like integrity, authentication and non-repudiation.

SAODV assumes that each ad hoc node has a signature key pair from a suitable asymmetric cryptosystem. Further, each node is capable of securely verifying the association between the address of other node and the public key of that node. A key management
scheme is needed for SAODV. Two mechanisms are used to secure the AODV messages:

- Digital signatures to authenticate the non-mutable fields of the messages, and
- Hash chains to secure the mutable hop count field of the message.

For the non-mutable fields, authentication can be performed in a point-to-point manner, but the techniques cannot be applied to the mutable information. RERR (Route Error) messages are protected in a different manner because of a big amount of mutable information. According to the author [15], [100], it is not important which node started the route error and which nodes are just forwarding it. The important information is that a neighbor node is informing other nodes that it is not able to route messages to certain destinations anymore. Therefore, every node (generating or forwarding a route error message) uses digital signatures to sign the whole RERR message and that any neighbor that receives RERR verifies the signature. The RREQ and RREP have the following extension fields:

\[
<\text{Type, Length, Hash function, Max Hop Count, Top hash, Signature, Hash}>
\]

The RERR has the following extension fields:

\[
<\text{Type, Length, Reserved, Signature}>
\]
**SAODV hash chains**

Hash chain is used to check the integrity of the hop count field of RREQ and RREP messages by allowing every node that receives the message to verify that the hop count has not been modified by malicious nodes. A hash chain is formed by repeatedly applying a one-way hash function to a seed number.

Every time a node initiates a RREQ or a RREP message, it performs the following operations [2]:

Generates a random number (seed).

Sets the $Max\_Hop\_Count$ field to the $TimeToLive$ value (from the IP header).

$Max\_Hop\_Count = TimeToLive$

Sets the Hash field to the seed value.

$Hash = seed$

Sets the $Hash\_Function$ field to the identifier of the hash function that it is going to use. The possible values are shown in Table 2.1.

<table>
<thead>
<tr>
<th>Value</th>
<th>Hash function</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Reserved</td>
</tr>
<tr>
<td>1</td>
<td>MD5HMAC96</td>
</tr>
<tr>
<td>2</td>
<td>SHA1HMAC96</td>
</tr>
<tr>
<td>3 – 127</td>
<td>Reserved</td>
</tr>
<tr>
<td>128 – 255</td>
<td>Implementation dependent</td>
</tr>
</tbody>
</table>

*Table 2.1 Possible values of the Hash function field*
Calculates $Top\_Hash$ by hashing seed $Max\_Hop\_Count$ times.

$$Top\_Hash = h^{Max\_Hop\_Count}(seed)$$

Where:

- $h$ is a hash function.
- $h^i(x)$ is the result of applying the function $h$ to $x$, $i$ times.

Every time a node receives a RREQ or a RREP message, it performs the following operations in order to verify the hop count:

Applies the hash function $h^{Max\_Hop\_Count - Hop\_Count}$ times to the value in the $Hash$ field, and verifies that the resultant value is equal to the value contained in the Top Hash field.

$$Top\_Hash = h^{Max\_Hop\_Count - Hop\_Count}(Hash)$$

Before re-broadcasting a RREQ or forwarding a RREP, a node applies the hash function to the $Hash$ value in the Signature Extension to account for the new hop $Hash = h(Hash)$

The $Hash\_Function$ value indicates which hash function has to be used to compute the hash. $Hash\_Function$, $Max\_Hop\_Count$, $Top\_Hash$, and $Hash$ fields are transmitted in the Signature Extension. And, as it will be explained in the next subsection, all of them but the $Hash$ field are signed to protect the origin integrity.

**SAODV digital signatures**

Digital signature is used to protect the integrity of the non-mutable data in RREQ and RREP messages. That is, every field except the $Hop\_Count$ of the AODV message and the $Hash$ from the SAODV
extension are signed. The issue in this work is that, AODV allows intermediate nodes to reply to a RREQ messages if they have a cached route to that destination. This feature provides more efficiency to this protocol, but it also makes it more difficult to secure. The RREP message generated by an intermediate node should be signed on behalf of the final destination. The author [67], [73] of this work presents different solutions to this problem:

[a] If an intermediate node cannot properly sign its RREP message, it just ignores the RREQ as if it does not have the cached route and forwards the RREQ message.

[b] Every time a node generates a RREQ message, it also includes the RREP flags, the prefix size and the signature that can be used by any intermediate node (which creates a reverse route to the originator of the RREQ) to reply to the RREQ message. Moreover, when an intermediate node generates a RREP message, the lifetime of the route has changed from the original one. Hence, the intermediate node should include both lifetimes (the old one is needed to verify the signature of the route destination) and sign the new lifetime. The original information of the route is signed by the final destination and the lifetime is signed by the intermediate node. To distinguish the different SAODV extension messages, the ones that have two signatures are called RREQ and RREP Double Signature Extension.
Upon receiving a RREQ message, a node first verifies the signature before creating or updating a reverse route to the source of the RREQ. If the RREQ was received with a Double Signature Extension, then the node will also store the signature for the RREP and the lifetime (which is the ‘reverse route lifetime’ value) in the route entry. An intermediate node will reply to a RREQ with a RREP only if it fulfills the AODV’s requirements and the node has the corresponding signature and old lifetime to put into the Signature and Old Lifetime fields of the RREP Double Signature Extension. Otherwise, it will rebroadcast the RREQ as it has no cached route. When the destination receives a RREQ, it will reply with a RREP with a Single Signature Extension. When a node receives a RREP, it first verifies the signature before creating or updating a route to that host. If the signature verification is successful, it will store the route with the signature of the RREP and the lifetime, otherwise the RREP is discarded.

2.3 SUMMARY OF THIS CHAPTER

This chapter provides an overview of the security issues in MANETs. It classifies the attacks that are possible against the existing routing protocols. An understanding of these attacks and their impacts on the routing mechanism will help researchers in designing secure routing protocols. This chapter also gives a brief survey of two secure routing protocols for MANETs – ARIADNE and SAODV. These protocols are just two among various solutions that have been proposed and researched. It is obvious that most of these secure
protocols are based upon the existing routing protocols. By adding security “layers”, they bear additional performance costs when compared to the underlying protocols. Some performance evaluations of these protocols have been conducted to better understand the tradeoffs between performance and security [17], [44], [66]. However, more research is needed to understand the performance of these secure protocols in the “real” environments with various exploits and malicious actions. Chapters IV and V will discuss more about these issues, and present a set of experiments that are designed to evaluate the performance of secure routing protocols in malicious scenarios.