Sensor-Based Intrusion Detection for Intra-Domain Distance-Vector Routing

Vishal Mittal, Giovanni Vigna
Reliable Software Group
University of California Santa Barbara
{vishal,vigna}@cs.ucsb.edu

ABSTRACT

Detection of routing-based attacks is difficult because malicious routing behavior can be identified only in specific network locations. In addition, the configuration of the signatures used by intrusion detection sensors is a time-consuming and error-prone task because it has to take into account both the network topology and the characteristics of the particular routing protocol in use. We describe an intrusion detection technique that uses information about both the network topology and the positioning of sensors to determine what can be considered malicious in a particular place of the network. The technique relies on an algorithm that automatically generates the appropriate sensor signatures. This paper presents a description of the approach, applies it to an intra-domain distance-vector protocol and reports the results of its evaluation.

Categories and Subject Descriptors

C.2.2 [**Computer-Communication Networks**]: Network Protocols—*Routing protocols,Protocol Verification*

General Terms

Security

Keywords

Routing Security, Intrusion Detection, Network Topology

1. INTRODUCTION

Attacks against the IP routing infrastructure can be used to perform substantial *denial-of-service* attacks or as a basis for more sophisticated attacks, such as *man-in-the-middle* and *non-blind-spoofing* attacks. Given the insecure nature of the routing protocols currently in use, preventing these attacks requires modifications to the routing protocols, the routing software, and, possibly, the network topology itself. Because of the critical role of routing, there is a considerable inertia in this process. As a consequence, insecure routing protocols are still widely in use throughout the Internet.

A complementary approach to securing the routing infrastructure relies on *detection* of routing attacks and execution of appropriate countermeasures. Detecting routing attacks is a complex task because malicious routing behavior can be identified only in specific network locations. In addition, routing information propagates from router to router, throughout the network. Therefore, the presence of malicious routing information is not necessarily restricted to the location where an attack is carried out.

We describe a misuse detection technique that uses a set of *sensors* deployed within the network infrastructure. Sensors are intrusion detection components equipped with a set of *signatures*, which describe the characteristics of malicious behavior. The traffic that is sent on the network links is matched against these signatures to determine if it is malicious or not.

The use of multiple sensors for intrusion detection is a well-established practice. The analysis of network traffic at different locations in the network supports more comprehensive intrusion detection with respect to single-point analysis. The disadvantage of a distributed approach is the difficulty of configuring the sensors according to the characteristics of the protected network. This problem is exacerbated by the nature of routing. The configuration of the sensors has to take into account the network topology, the positioning of the sensors in the network, and the characteristics of the particular routing protocol in use. In addition, some attacks can be detected only by having sensors communicate with each other. As a consequence, the configuration of the signatures used by the sensors is a time-consuming and error-prone task.

The novel contribution of our approach is an algorithm that, given a network topology and the positioning of the intrusion detection sensors, can automatically determine both the signature configuration of the sensors and the messages that the sensors have to exchange to detect attacks against the routing infrastructure. This paper introduces the general approach and describes its application to the *Routing Information Protocol (RIP)*.

RIP is an intra-domain distance-vector routing protocol [13]. At startup, every RIP router knows only its own addresses and the links corresponding to these addresses. Every RIP router propagates this information to its immediate neighbors. On receiving the routing information, the neighbors update their routing tables to add, modify, or delete routes to the advertised destinations.

Routers add a route to a destination if they do not have one. A route is modified if the advertised route is better than the one that the router already has. If a router receives a message from a neighbor advertising *unreachability* to a certain destination and if the router is using that neighbor to reach the destination, then the router deletes the route

CCS'02, November 18–22, 2002, Washington, DC, USA.

to the destination from its routing table.

Under certain conditions, RIP might not converge. It may exhibit the *bouncing-effect* problem or the *count-to-infinity* problem [9]. These problems are partially overcome by using the *split-horizon* technique, *triggered-updates*, and by limiting the number of hops that can be advertised for a destination [1].

In order to decide whether a routing update is malicious or not, a router needs to have reliable, non-local topology information. Unfortunately, RIP routers do not have this information. To support router-based intrusion detection it would be necessary to modify both the RIP protocol and the routing software. Therefore, our approach relies on external sensors.

Sensor configurations are generated offline on the basis of the complete network topology and the positions of the sensors in the network. The configuration generation algorithm determines every possible path from every router to every other router. The configuration for an individual sensor is a subset of this information based on the position of the sensor in the network. Sensors need to be reconfigured if routers and links are added to the topology. However, sensors do not need to be reconfigured if the topology changes due to link or router failures.

The remainder of this paper is organized as follows. Section 2 discusses related work in the field. Section 3 introduces an abstract reference model of the network routing infrastructure. Section 4 presents an algorithm to generate the configuration of intrusion detection sensors for the RIP distance-vector protocol. Section 5 discusses how routing attacks are detected. Section 6 describes the experimental setup that was used to analyze the attacks and evaluate the detection technique. Section 7 discusses the strengths and weaknesses of the approach. Section 8 draws some conclusions and outlines future work.

2. RELATED WORK

One of the earliest works on securing routing protocols is by Radia Perlman [16]. Perlman suggests the use of digital signatures in routing messages to protect against Byzantine failures. The main drawback of this approach is that generating digital signatures is a computationally intensive task. Signature verification is usually not as expensive, but most of the solutions that use digital signatures require that a number of them be verified, leading to a considerable performance overhead.

The use of digital signatures is also advocated by Murphy *et al.* [14, 15] for both distance-vector and link-state advertisements. Kent *et al.* [12, 11] describe an approach to allow the recipient of a BGP [18] message to verify the entire path to a destination. Smith *et al.* [19] introduce a scheme to protect BGP using digital signatures and also describe a scheme to secure distance-vector routing protocols by using predecessor information [20].

Several other schemes have been proposed to reduce the performance overhead associated with securing routing protocols using digital signatures. Hauser *et al.* [7] describe two techniques for efficient and secure generation and processing of updates in link-state routing. Zhang [24] describes that routing messages can be protected by using *one-time signatures* on message chains. In [6], Goodrich describes a *leap-frog* cryptographic signing protocol that uses secret-key cryptography.

While the research referenced so far focuses on preventing attacks, a complementary approach to the problem of securing the routing infrastructure focuses on detecting attacks [1, 5, 8]. For example, Cheung *et al.* [3, 4] present solutions to the *denial-of-service* problem for the routing infrastructure using intrusion detection. Another example is a protocol called *WATCHERS* described by Bradley *et al.* [2]. The protocol detects and reacts to routers that drop or misroute packets by applying the principle of conservation of flow to the routers in a network. The *JiNao* project at MCNC/NCSU focuses on detecting intrusions, especially insider attacks, against OSPF. Wu *et al.* [21, 17, 23, 22, 10] consider how to efficiently integrate security control with intrusion detection in a single system.

The approach described in this paper differs from other intrusion detection approaches because it focuses on the topological characteristics of the network to be protected and the placement of the detection sensors. The approach relies on topology information to automatically generate the signatures used by sensors to detect the attacks. The details of the algorithm and its application are described in the following sections.

3. REFERENCE MODEL

An abstract reference model of the network is introduced to describe the algorithm used to generate the signatures and how these signatures are used to detect attacks against the routing infrastructure. A network is represented by an undirected graph $G = (V, E)$ where vertices $V = \{v_1, v_2, \cdots, v_n\}$ denote the set of routers. Positive weight edges $E = \{e_1, e_2, \cdots, e_m\}$ represent the links connecting router interfaces. An edge $e_{ij} \in E$ connects routers v_i and v_j.[2]

A *subnet* is a range of IP addresses with the same network mask. Every link e_{ij} is associated with a subnet s_{ij}. $S = \{s_1, s_2, \cdots, s_m\}$ is the set of subnets corresponding to the set of links E. We assume that $e_{ij} = e_{ji}$ and $s_{ij} = s_{ji}$. Every vertex $v_j \in V$ has an associated set $E_j = \{e_{j1}, e_{j2}, \cdots\} \subset E$ that represents the set of edges connected to v_j. $S_j = \{s_{j1}, s_{j2}, \cdots\}$ is the set of subnets corresponding to the set of links E_j. Every link e_{ij} is associated with a cost c_{ij}, with $c_{ij} = c_{ji}$.[3] A sensor placed on link e_{ij} is identified as $sensor_{ij}$. A host p that is connected to link e_{ij} is denoted by h^p_{ij}.

Consider the sub-graph $G_{sub} \subset G$ shown in Figure 1. In that context, routing problems can occur due to faults in routers v_i, v_j, and v_k or due to malicious actions of both the routers and the host h^p_{ij}. Hosts or routers are termed malicious when they have been compromised and are used as a means to modify the normal operation of the network. Both network failures and threats due to malicious intent need to be included in the threat model because a sensor cannot always differentiate between the two. The approach described here is concerned with attacks involving injection, alteration, or removal of routes to specific destinations, by

[1] Limiting the number of hops ensures that a route is declared unusable, when the protocol does not converge and the number of advertised hops exceeds the maximum number of allowed hops. However, this also limits the diameter of the networks in which RIP can be used.

[2] For the sake of simplicity, this model does not consider the possibility of more than two routers being connected to a single link. The model can be extended to support that possibility by assuming graph G to be a hyper-graph.

[3] The model does not consider the possibility of links being asymmetric, i.e., having different costs in different directions. The model can be extended to support asymmetric costs by assuming graph G to be a directed graph where $e_{ij} \neq e_{ji}$, and $c_{ij} \neq c_{ji}$.

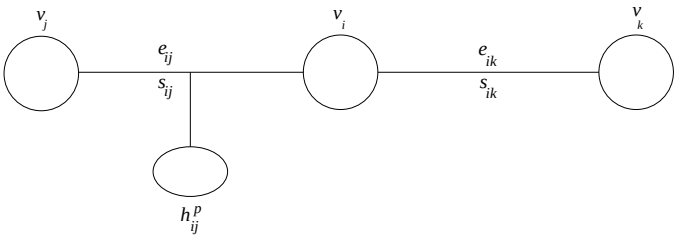

Figure 1: Threat Reference Model

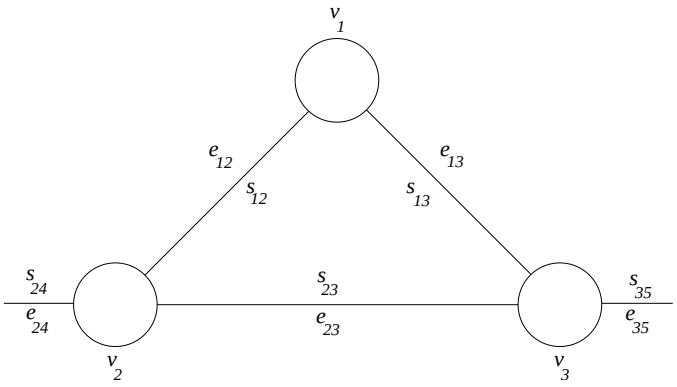

Figure 2: Sensor Configuration Example

malicious hosts or routers. Security threats, e.g., unauthorized access to information, dropping and alteration of data packets by routers, etc., are not addressed here. More precisely, we can describe our threat model with respect to G_{sub} as follows:

1. v_i fails. This will result in sub-optimal paths or no paths at all from s_{ij} to s_{ik} and vice-versa.

2. v_i is compromised and misconfigured to advertise a sub-optimal or *infinite-cost* path to s_{ik}. This will result in sub-optimal paths or no paths at all from s_{ij} to s_{ik}.

3. v_j is compromised and misconfigured to advertise a better than optimal path for s_{ik}. This will result in hosts from subnet s_{ij} using v_j to reach hosts in subnet s_{ik} even though v_i has a better route to subnet s_{ik}. If v_j actually has no path to subnet s_{ik} then packets from subnet s_{ij} will not reach subnet s_{ik}.

4. h^p_{ij} pretends to be v_i with respect to v_j or pretends to be either v_j or v_k with respect to v_i. h^p_{ij} can then advertise routes as in 2 and 3.

5. h^p_{ij} renders v_i unusable and pretends to be v_i. h^p_{ij} advertises the same information as v_i but since v_i is unusable, packets do not get routed.

G_{sub} represents a segment of the network, represented by G, where incorrect routing information is generated. If there is a sensor present on every link of G_{sub}, the faulty or malicious entity can be identified and a response process can be initiated. In the absence of sensors on every link of G_{sub}, the incorrect routing information can propagate to the rest of the network. In this case, the attack can still be detected, but determining the origin of the attack requires an analysis of the effects of the incorrect routing information. This analysis is considerably difficult to perform and is not in the scope of the approach described here.

4. SENSOR CONFIGURATION

A sensor is configured on the basis of the network topology and the sensor's position in the network. A sensor's position in the network is specified by the link on which the sensor is placed. A separate component, called the *Sensor Configurator*, is given the network topology and the position of all available sensors as inputs. The Sensor Configurator uses an algorithm to generate the configuration of each sensor. The configurations are then loaded into the appropriate sensors.

The first step of the Sensor Configurator algorithm is to find all paths and their corresponding costs from every router to every other router in the graph G. The results are organized into a 2-dimensional vertex-to-vertex matrix. The $(i,j)^{th}$ entry of the matrix points to a list of 2-tuples $\{(c^k_{ij}, p^k_{ij})\}$. The list contains a tuple for each path between vertices v_i and v_j. p^k_{ij} is a set of vertices traversed to reach v_j from v_i. c^k_{ij} is the cost of the path p^k_{ij} between vertices v_i and v_j. For example, consider Figure 2. v_1, v_2, v_3 are routers. e_{12}, e_{13}, e_{23}, e_{24}, e_{35} are links with associated subnets s_{12}, s_{13}, s_{23}, s_{24}, s_{35}, respectively. The cost of all links is equal to 1. Table 1 shows the vertex-to-vertex matrix for the graph in Figure 2. The $\{$*(cost, path)*$\}$ entry in row v_1, column v_1, is the set of possible paths and their corresponding costs that vertex v_1 can take to reach vertex v_1. The entry $\{(0, (\emptyset))\}$ means that vertex v_1 can reach vertex v_1 at cost 0, through itself. The entry in row v_1, column v_2, means that vertex v_1 can reach vertex v_2 at cost 1 through itself or at cost 2 through vertex v_3. In the second step of the algorithm, the vertex-to-vertex matrix is transformed into a 2-dimensional vertex-to-subnet matrix. Each column, representing a vertex v_j of the vertex-to-vertex matrix, is replaced by a set of columns, one for each subnet directly connected to the vertex, i.e., one for each member of S_j. Consider the set of columns $S_j = \{s_{jx}, s_{jy}, s_{jz}, \cdots\}$ replacing column v_j. The set of paths in the vertex-to-subnet matrix is $\{(p^k_{i,jx} = (p^k_{ij}, v_j)), (p^k_{i,jy} = (p^k_{ij}, v_j)), (p^k_{i,jz} = (p^k_{ij}, v_j)), \cdots\}$ where $p^k_{i,jx}$ is the k^{th} path from router v_i to subnet s_{jx} in the vertex-to-subnet matrix and p^k_{ij} is the corresponding path in the vertex-to-vertex matrix. The set of costs in the vertex-to-subnet matrix is $\{(c^k_{i,jx} = c^k_{ij} + c_{jx}), (c^k_{i,jy} = c^k_{ij} + c_{jy}), (c^k_{i,jz} = c^k_{ij} + c_{jz}), \cdots\}$ where $c^k_{i,jx}$ is the cost of the k^{th} path from router v_i to subnet s_{jx} in the vertex-to-subnet matrix and c^k_{ij} is the corresponding cost in the vertex-to-vertex matrix. c_{jx} is the cost of the link e_{jx} associated with subnet s_{jx}. This cost must be taken into account because c^k_{ij} only represents the cost of the path from router v_i to v_j. The cost to reach subnet s_{jx} from router v_j should be added to c^k_{ij} to get the total cost $c^k_{i,jx}$. For example, the vertex-to-vertex matrix shown in Table 1 is transformed into the vertex-to-subnet matrix shown in Table 2. The $\{$*(cost, path)*$\}$ entry in row v_1, column s_{12} is the set of possible paths that vertex v_1 can take to reach subnet s_{12} and their corresponding costs. The entry $\{(1, (v_1))\}$ means that vertex v_1 can reach subnet s_{12} at cost 1, through v_1. The entry in row v_1, column s_{23}, means that vertex v_1 can reach subnet s_{23} at cost 2 through vertex v_2 or at cost 3 through vertices v_3 and v_2, and so on.

	v_1	v_2	v_3
v_1	$\{(0, (\emptyset))\}$	$\{(1, (\emptyset)), (2, (v_3))\}$	$\{(1, (\emptyset)), (2, (v_2))\}$
v_2	$\{(1, (\emptyset)), (2, (v_3))\}$	$\{(0, (\emptyset))\}$	$\{(1, (\emptyset)), (2, (v_1))\}$
v_3	$\{(1, (\emptyset)), (2, (v_2))\}$	$\{(1, (\emptyset)), (2, (v_1))\}$	$\{(0, (\emptyset))\}$

Table 1: Vertex-to-Vertex Matrix

	s_{12}	s_{13}	s_{21}	s_{23}	s_{24}	s_{31}	s_{32}	s_{35}
v_1	$\{(1, (v_1))\}$	$\{(1, (v_1))\}$	$\{(2, (v_2)),$ $(3, (v_3, v_2))\}$	$\{(2, (v_2)),$ $(3, (v_3, v_2))\}$	$\{(2, (v_2)),$ $(3, (v_3, v_2))\}$	$\{(2, (v_3)),$ $(3, (v_2, v_3))\}$	$\{(2, (v_3)),$ $(3, (v_2, v_3))\}$	$\{(2, (v_3)),$ $(3, (v_2, v_3))\}$
v_2	$\{(2, (v_1)),$ $(3, (v_3, v_1))\}$	$\{(2, (v_1)),$ $(3, (v_3, v_1))\}$	$\{(1, (v_2))\}$	$\{(1, (v_2))\}$	$\{(1, (v_2))\}$	$\{(2, (v_3)),$ $(3, (v_1, v_3))\}$	$\{(2, (v_3)),$ $(3, (v_1, v_3))\}$	$\{(2, (v_3)),$ $(3, (v_1, v_3))\}$
v_3	$\{(2, (v_1)),$ $(3, (v_2, v_1))\}$	$\{(2, (v_1)),$ $(3, (v_2, v_1))\}$	$\{(2, (v_2)),$ $(3, (v_1, v_2))\}$	$\{(2, (v_2)),$ $(3, (v_1, v_2))\}$	$\{(2, (v_2)),$ $(3, (v_1, v_2))\}$	$\{(1, (v_3))\}$	$\{(1, (v_3))\}$	$\{(1, (v_3))\}$

Table 2: Vertex-to-Subnet Matrix

The vertex-to-subnet matrix is in a format that is similar to the one used by the routers themselves.

Once the vertex-to-subnet matrix has been computed for the entire network, the portion of the vertex-to-subnet matrix relevant to each sensor is extracted. Each sensor uses its vertex-to-subnet matrix to validate the routing advertisements that are sent on the link on which the sensor is placed. More precisely, $sensor_{ij}$, placed on link e_{ij}, needs to validate routing advertisements from routers v_i and v_j only.[4] Therefore, the vertex-to-subnet matrix for $sensor_{ij}$ has rows from the common vertex-to-subnet matrix corresponding to v_i and v_j only.

Some entries in a vertex-to-subnet matrix may not correspond to actual routing information. Therefore, the matrix can be further reduced by ignoring these entries. Consider the vertex-to-subnet matrix for $sensor_{ij}$ on link e_{ij} with subnet s_{ij} between routers v_i and v_j. s_{ik} is the subnet on link e_{ik} between routers v_i and v_k. s_{ab} is any other subnet. In this context, the vertex-to-subnet matrix for $sensor_{ij}$ is reduced according to the following rules:

1. For neighboring routers v_i and v_k, row v_i, columns s_{ik} or s_{ki}, a $\{$*(cost, path)*$\}$ tuple is ignored if the path is of the form (v_k). For example, in Table 2, for row v_1, column s_{31}, the $\{$*(cost, path)*$\}$ tuple $\{(2, (v_3))\}$ is ignored because subnet s_{31} is directly connected to router v_1. Therefore, v_1 will never advertise a route on link e_{12} for subnet s_{31} through router v_3 at cost 2.

2. For any row, columns s_{ab} or s_{ba}, a $\{$*(cost, path)*$\}$ tuple is ignored if the path is of the form $(\cdots, v_a, v_b, \cdots)$ or $(\cdots, v_b, v_a, \cdots)$. For example, in Table 2, for row v_1, column s_{23}, the $\{$*(cost, path)*$\}$ tuple $\{(3, (v_3, v_2))\}$ is ignored because router v_1 can reach subnet s_{23} at cost 2 through router v_3. Therefore, v_1 will not advertise a route to subnet s_{23} through (v_3, v_2) at cost 3.

3. For neighboring routers v_i and v_j, row v_i, columns s_{ij} or s_{ji}, a $\{$*(cost, path)*$\}$ tuple is ignored. For example, in Table 2, for row v_1, column s_{12}, the $\{$*(cost, path)*$\}$ tuple $\{(2, (v_2))\}$ is ignored because both routers v_1 and v_2 are directly connected to subnet s_{12} and have the same cost to s_{12}. Router v_2 will never use a longer path through v_1 to reach a directly connected subnet s_{12}. Therefore, v_1 will never advertise such a route to v_2.

4. For neighboring routers v_i and v_j, row v_i, any column, a $\{$*(cost, path)*$\}$ tuple is ignored if the path is of the form $(v_j, \cdots)$. For example, in Table 2, for row v_1, column s_{23}, the $\{$*(cost, path)*$\}$ tuple $\{(2, (v_2))\}$ is ignored because if in the route advertised by v_1 for subnet s_{23} the first hop is router v_2, then v_1 has learned that route from v_2. This implies that v_2 has a better route to s_{23} than v_1 has and will never use v_1 to reach s_{23}. Therefore, v_1 will never advertise such a route to v_2. The *split-horizon* check in RIP ensures the same thing.

After the simplification of the vertex-to-subnet matrix for $sensor_{ij}$, with rows v_i and v_j, the term v_i is actually replaced by a tuple $\{v_i^{link}, v_i^{ip}\}$ where v_i^{link} is the link-level address of the interface of router v_i that is connected to e_{ij} and v_i^{ip} is the corresponding IP address. Similarly, the term v_j is replaced by a tuple $\{v_j^{link}, v_j^{ip}\}$. Finally, the information regarding the position of other sensors is added to the vertex-to-subnet matrix by marking the links where the sensors are placed.

5. SENSOR DETECTION ALGORITHM

Once the offline process of generating the sensor configurations is completed, the configurations are loaded into the sensors. At run-time the sensors analyze the routing advertisements that are sent on the corresponding link. They match the contents of a routing advertisement with their configuration to decide whether the routing advertisement represents evidence of an attack or not.

Consider $sensor_{ij}$, placed on link e_{ij}. In addition to storing v_i^{link} and v_i^{ip} for router v_i and v_j^{link} and v_j^{ip} for router v_j, $sensor_{ij}$ also stores $e_{ij}^{link_{bcast}}$ and $e_{ij}^{ip_{bcast}}$, which are the link-level and IP broadcast addresses for link e_{ij} and $rip^{link_{mcast}}$ and $rip^{ip_{mcast}}$, which are the link-level and IP multicast addresses for RIP routers.

In its vertex-to-subnet matrix, $sensor_{ij}$ also stores $\{$*(cost, path)*$\}$ sets from router v_i to subnet s_{ab} of the form $\{(c_{i,ab}^{o1}, p_{i,ab}^{o1}), (c_{i,ab}^{o2}, p_{i,ab}^{o2}), \cdots, (c_{i,ab}^{s1}, p_{i,ab}^{s1}), (c_{i,ab}^{s2}, p_{i,ab}^{s2}), \cdots\}$. $c_{i,ab}^{o1}$ is the optimal cost at which router v_i can send data to subnet s_{ab}, through path $p_{i,ab}^{o1}$. There can be multiple optimal-cost paths $\{p_{i,ab}^{o1}, p_{i,ab}^{o2}, \cdots\}$ with costs $\{c_{i,ab}^{o1}, c_{i,ab}^{o2}, \cdots\}$ from router v_i to subnet s_{ab} such that $c_{i,ab}^{o1} = c_{i,ab}^{o2} = \cdots = c_{i,ab}^{o}$. Router v_i can also send data to subnet s_{ab} through a path

[4] A sensor needs to validate routing advertisements only from routers connected to the link on which it is placed, because distance-vector routers advertise routing information only on links connected to them directly.

$p^{s_1}_{i,ab}$ with a sub-optimal cost $c^{s_1}_{i,ab}$. There can be multiple sub-optimal-cost paths $\{p^{s_1}_{i,ab}, p^{s_2}_{i,ab}, \cdots\}$ with costs $\{c^{s_1}_{i,ab}, c^{s_2}_{i,ab}, \cdots\}$ from router v_i to subnet s_{ab}.

Next, consider a distance-vector routing advertisement m, where m is of the type:

[Link-Level-Header [IP-Header [UDP-Header [Distance-Vector Routing Information]]]]

For routing advertisement m, $m^{link_{src}}$ and $m^{link_{dst}}$ are the link-level source and destination addresses respectively, $m^{ip_{src}}$ and $m^{ip_{dst}}$ are the IP source and destination addresses respectively, m^{ttl} is the *time-to-live* field in the IP header, and $m^{c_{ab}}$ is the cost advertised for subnet $m^{s_{ab}}$. By using the information stored by the sensor and the information contained in the routing message it is possible to verify the correctness of link-level and network-level information, the plausibility of the distance-vector information, the messages that are needed to verify advertised routes, and the frequency of routing updates. These four verifications are described in the following sections.

5.1 Link-Level and Network-Layer Information Verification

A legitimate routing advertisement must have the link-level address and IP address of one of the routers connected to the link[5] and have a time-to-live value equal to 1.

The following is a relation between the fields $m^{link_{src}}$, $m^{link_{dst}}$, $m^{ip_{src}}$, $m^{ip_{dst}}$, m^{ttl}, $m^{c_{ab}}$ and $m^{s_{ab}}$, of a legitimate routing advertisement m:

$$
\begin{array}{llll}
\{[(m^{link_{src}} = v_i^{link} & \wedge & m^{ip_{src}} = v_i^{ip}) & \wedge \\
\{(m^{link_{dst}} = v_j^{link} & \wedge & m^{ip_{dst}} = v_j^{ip}) & \vee \\
(m^{link_{dst}} = e_{ij}^{link_{bcast}} & \wedge & m^{ip_{dst}} = e_{ij}^{ip_{bcast}}) & \vee \\
(m^{link_{dst}} = rip^{link_{mcast}} & \wedge & m^{ip_{dst}} = rip^{ip_{mcast}})\}] & \\
 & \vee & & \\
[(m^{link_{src}} = v_j^{link} & \wedge & m^{ip_{src}} = v_j^{ip}) & \wedge \\
\{(m^{link_{dst}} = v_i^{link} & \wedge & m^{ip_{dst}} = v_i^{ip}) & \vee \\
(m^{link_{dst}} = e_{ji}^{link_{bcast}} & \wedge & m^{ip_{dst}} = e_{ji}^{ip_{bcast}}) & \vee \\
(m^{link_{dst}} = rip^{link_{mcast}} & \wedge & m^{ip_{dst}} = rip^{ip_{mcast}})\}] & \\
 & \wedge & & \\
(m^{ttl} = 1) & & &
\end{array}
$$

In the above relation, if the link-level and IP source addresses of routing advertisement m are those of router v_i, then the link-level and IP destination addresses of m should be those of router v_j, the broadcast address of link e_{ij}, or the multicast address of RIP routers. If m has originated from router v_j, the source link-level and IP addresses should be those of router v_j and destination link-level and IP addresses should be those of router v_i, the broadcast address of link e_{ij}, or the multicast address of RIP routers. The time-to-live field of m should be 1. Note that link-level and network-layer information can be spoofed. Therefore, this verification alone is not enough to conclude that a routing advertisement is not malicious.

5.2 Distance-Vector Information Verification

Routing advertisements for subnets that do not belong to the local autonomous system indicate that the routing advertisements are malicious.[6] A sensor scans a routing advertisement for attacks by matching every advertised subnet against the list of subnets that the sensor has in its configuration. If the sensor does not find a match, it declares the routing advertisement to be malicious.

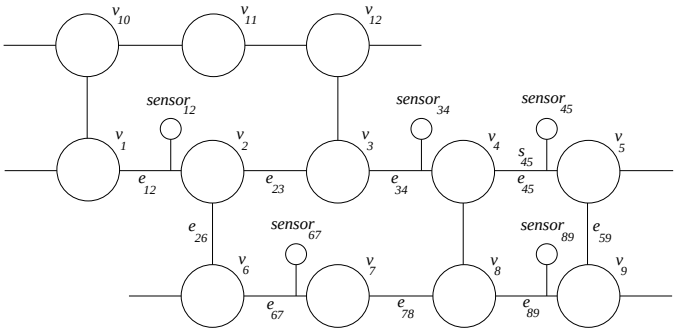

Figure 3: Sensor Detection Example

Next, a routing advertisement is analyzed to determine whether the advertised cost is *optimal*, *sub-optimal*, or *impossible*. An optimal cost is the cost of one of the shortest paths from a router to a destination advertised by the router. A sub-optimal cost is the cost of one of the paths from a router to a destination advertised by the router. This path may not necessarily be the shortest. An impossible cost is a cost not associated to any of the paths from a router to a destination advertised by the router. *Unreachability*, i.e., where the advertised cost is 15, is considered sub-optimal rather than impossible.

Consider a routing advertisement m, originating from router v_i on link e_{ij}, advertising a cost $m^{c_{ab}}$ for subnet $m^{s_{ab}}$. The sensor configuration defines the following costs for reaching s_{ab} from v_i:

$$
\begin{array}{ll}
m^{c_{ab}} \in \{c^{o_1}_{i,ab}, c^{o_2}_{i,ab}, \cdots\} & \textit{optimal-cost} \\
m^{c_{ab}} \in \{c^{s_1}_{i,ab}, c^{s_2}_{i,ab}, \cdots\} & \textit{sub-optimal-cost} \\
m^{c_{ab}} \notin \{c^{o_1}_{i,ab}, \cdots, c^{s_1}_{i,ab}, \cdots\} & \textit{impossible-cost}
\end{array}
$$

The above relation states that in a routing advertisement m, advertised by router v_i for subnet s_{ab}, the cost $m^{c_{ab}}$ is optimal if it belongs to the set of optimal costs; sub-optimal if it belongs to the set of sub-optimal costs; impossible if it does not belong to the set of optimal costs or sub-optimal costs. Assuming that a sensor has the correct topological information, an impossible-cost advertisement detected by the sensor is considered malicious. Better-than-optimal-cost routing advertisements can be detected by checking the routing advertisements for impossible-cost advertisements. For example, consider Figure 3. The costs of all links are equal to 1. The set of optimal costs that router v_2 can advertise for subnet s_{45}, on link e_{12}, is $\{3\}$, using path $\{(v_2, v_3, v_4)\}$. The set of sub-optimal costs that router v_2 can advertise for subnet s_{45}, on link e_{12}, is $\{5, 6, 6\}$ with paths $\{(v_2, v_6, v_7, v_8, v_4), (v_2, v_6, v_7, v_8, v_9, v_5), (v_2, v_3, v_4, v_8, v_9, v_5)\}$, respectively. No paths from router v_1 are considered, since router v_2 will not advertise any paths that it has learned through router v_1 on link e_{12}. Costs advertised by router v_2 on link e_{12} for subnet s_{45} that are not 3, 5, or 6 are impossible costs.

[5] In distance-vector routing protocols, routing advertisements are meant for a router's immediate neighbors only.

[6] Routers running intra-domain routing protocols do not have routes to every possible subnet. Usually, routes to subnets outside the autonomous system are only known to the border routers. Non-border routers use default routes to the border routers for external subnets.

5.3 Path Verification

The impossible-cost-advertisement verification cannot determine the validity of an optimal-cost or sub-optimal-cost advertisement. A malicious entity can advertise a sub-optimal cost to a destination even though the corresponding optimal-cost path is available for use. For example, in Figure 3, subnet s_{45} can be reached at cost 3 from router v_2 but a malicious entity on link e_{12} can pretend to be router v_2 and advertise cost 6 for subnet s_{45}. This will result in router v_1 using router v_{10}, instead of router v_2, to send data to subnet s_{45}. A malicious entity can also advertise an optimal cost when the optimal-cost path is not available. For example, in Figure 3, a malicious entity on link e_{12} can pretend to be router v_2 and advertise cost 3 for subnet s_{45} when subnet s_{45} is no longer reachable using path $\{(v_2, v_3, v_4)\}$. In addition, a malicious entity can subvert a router and configure it to drop data packets, or it may impersonate a router after having disabled the router.

In all the above attacks, the advertised distance-vector information is correct. Therefore, these attacks cannot be detected by merely verifying the distance-vector information. To detect such attacks, sensors use a path-verification algorithm. Consider $sensor_{ij}$ on link e_{ij} between routers v_i and v_j. If $sensor_{ij}$ finds in a routing advertisement m the cost $m^{c_{ab}}$ advertised by router v_i for subnet $m^{s_{ab}}$ to be optimal or sub-optimal, then for all costs $c^k_{i,ab} \in \{c^{o1}_{i,ab}, c^{o2}_{i,ab}, \cdots, c^{s1}_{i,ab}, c^{s2}_{i,ab}, \cdots\}$ where $c^k_{i,ab} = m^{c_{ab}}$, the sensor searches its configuration for all paths $p^k_{i,ab}$ that have cost $c^k_{i,ab}$.

The set of sensors on path $p^k_{i,ab}$ is $Sensor^k_{i,ab}$. $sensor_{ij}$ verifies path $p^k_{i,ab}$ by sending a message to every $sensor_{yz} \in Sensor^k_{i,ab}$. For example, consider Figure 3. In this case, $sensor_{12}$ on link e_{12} detects a routing advertisement from router v_2 for subnet s_{45} at cost 3. Therefore, $sensor_{12}$ searches its configuration to find all paths that have cost 3. $sensor_{12}$ finds that path $\{(v_2, v_3, v_4)\}$ has cost 3. In its configuration, $sensor_{12}$ also has the information that for path $\{(v_2, v_3, v_4)\}$, links e_{34} and e_{45} have $sensor_{34}$ and $sensor_{45}$ on them. To validate the advertisement, $sensor_{12}$ sends messages to $sensor_{34}$ and $sensor_{45}$. If the advertised cost had been 5, $sensor_{12}$ would have sent messages to $sensor_{67}$, $sensor_{89}$ and $sensor_{45}$.

The path-verification algorithm can be more formally stated as follows. $sensor_{ij}$ on link e_{ij} verifies a routing advertisement m from router v_i, advertising an optimal or sub-optimal cost $m^{c_{ab}}$ for subnet s_{ab}, using the following steps:

1. If $m^{c_{ab}}$ is the optimal cost $c^o_{i,ab}$ from router v_i to subnet s_{ab}, $sensor_{ij}$ searches its configuration to find all paths $p^k_{i,ab}$ from router v_i to subnet s_{ab}, corresponding to cost $c^o_{i,ab}$, and the set of available sensors $Sensor^k_{i,ab}$ on those paths.

2. If $Sensor^k_{i,ab} = \{\emptyset\}$, i.e., there are no sensors on path $p^k_{i,ab}$, then $sensor_{ij}$ cannot verify if a path from router v_i to subnet s_{ab} exists. If there are sensors on every link of path $p^k_{i,ab}$ then the entire path can be verified. If sensors are not present on every link of path $p^k_{i,ab}$ but a sensor is present on e_{ab} then the intermediate path cannot be verified but it can be verified that subnet s_{ab} is reachable from router v_i. If there is no sensor present on e_{ab} then it cannot be verified whether subnet s_{ab} is reachable from router v_i or not.

3. If $Sensor^k_{i,ab} \neq \{\emptyset\}$ then $sensor_{ij}$ sends a message to every $sensor_{yz} \in Sensor^k_{i,ab}$ for every path $p^k_{i,ab}$.

4. Every path $p^k_{i,ab}$ is an available path for which every $sensor_{yz} \in Sensor^k_{i,ab}$ replies to $sensor_{ij}$. If there are one or more available paths $p^k_{i,ab}$, m is considered a valid routing advertisement. If there are none, $sensor_{ij}$ declares m to be malicious.

5. If $m^{c_{ab}}$ is a sub-optimal cost, $sensor_{ij}$ searches its configuration to find paths $p^{q_k}_{i,ab}$ from router v_i to subnet s_{ab} corresponding to every cost $c^q_{i,ab}$ such that $c^o_{i,ab} \leq c^q_{i,ab} \leq m^{c_{ab}}$. $c^o_{i,ab}$ is the optimal cost from router v_i to subnet s_{ab}. $sensor_{ij}$ also determines the sets of available sensors $Sensor^{q_k}_{i,ab}$ corresponding to paths $p^{q_k}_{i,ab}$.

6. $sensor_{ij}$ sends a message to every $sensor_{yz} \in Sensor^{s_k}_{i,ab}$ for paths $p^{s_k}_{i,ab}$ from router v_i to subnet s_{ab} corresponding to every cost $c^s_{i,ab}$ such that $c^o_{i,ab} \leq c^s_{i,ab} < m^{c_{ab}}$, where $c^o_{i,ab}$ is the optimal cost from router v_i to subnet s_{ab}. Note that the only difference between $p^{q_k}_{i,ab}$ and $p^{s_k}_{i,ab}$ is that the latter does not contain paths with costs equal to $m^{c_{ab}}$. Therefore, $p^{s_k}_{i,ab} \subset p^{q_k}_{i,ab}$ and $Sensor^{s_k}_{i,ab} \subset Sensor^{q_k}_{i,ab}$.

7. Every path $p^{s_k}_{i,ab}$ is an available path for which every $sensor_{yz} \in Sensor^{s_k}_{i,ab}$ replies to $sensor_{ij}$. If no available paths $p^{s_k}_{i,ab}$ exist, then $sensor_{ij}$ verifies paths $p^k_{i,ab}$ from router v_i to subnet s_{ab} corresponding to cost $m^{c_{ab}}$. Every path $p^k_{i,ab}$ for which every $sensor_{yz} \in Sensor^k_{i,ab}$ replies to $sensor_{ij}$ is an available path. If there are one or more available paths $p^k_{i,ab}$, then routing advertisement m with cost $m^{c_{ab}}$ is considered a valid routing advertisement. If there are no available paths then $sensor_{ij}$ declares m to be malicious.

8. For every available path $p^{s_k}_{i,ab}$, i.e., where every $sensor_{yz} \in Sensor^{s_k}_{i,ab}$ replies to $sensor_{ij}$, $sensor_{ij}$ waits for a time-period t_{delay}. For every available path $p^{s_k}_{i,ab}$, if $sensor_{ij}$ does not get a routing advertisement m', with cost $c^{s_k}_{i,ab}$, within t_{delay}, then $sensor_{ij}$ declares routing advertisement m to be malicious.

For example, consider Figure 3. $sensor_{12}$ detects a routing advertisement from router v_2 for subnet s_{45} at cost 3. $sensor_{12}$ searches its configuration and finds 3 to be the optimal cost from router v_2 to subnet s_{45}. $sensor_{12}$ finds path $\{(v_2, v_3, v_4)\}$ that has cost 3. Therefore, $sensor_{12}$ sends messages to sensors available on this path, i.e., $sensor_{34}$ and $sensor_{45}$. If $sensor_{12}$ does not get replies from both $sensor_{34}$ and $sensor_{45}$, it concludes that the path from router v_2 to subnet s_{45} is unavailable. If $sensor_{12}$ gets a reply from $sensor_{45}$ but not from $sensor_{34}$, $sensor_{12}$ can conclude that subnet s_{45} is reachable from router v_2 but it cannot verify the path. If $sensor_{12}$ gets a reply from $sensor_{34}$ but not from $sensor_{45}$, $sensor_{12}$ cannot be sure if subnet s_{45} is reachable from router v_2 or not. If $sensor_{12}$ gets a reply from both $sensor_{34}$ and $sensor_{45}$, $sensor_{12}$ can be sure that subnet s_{45} is reachable from router v_2. For the placement of sensors in Figure 3, $sensor_{12}$ can never be sure of the complete optimal path from router v_2 to subnet s_{45}, since link e_{23} does not have a sensor on it.

Assume now that $sensor_{12}$ detects a routing advertisement from router v_2 for subnet s_{45} at cost 5. The sensor searches its configuration and finds 5 to be a sub-optimal cost from router v_2 to subnet s_{45} and identifies path $\{(v_2, v_6, v_7, v_8, v_4)\}$ as the only path that has cost 5 from router

v_2 to subnet s_{45}. In addition, the sensor looks for paths that have costs greater than or equal to the optimal cost and less than 5. The only such path in this case is the path having the optimal cost 3. Therefore, $sensor_{12}$ sends messages to sensors available on the path having cost 3. If $sensor_{12}$ finds this path unavailable then $sensor_{12}$ sends messages to sensors available on the path having cost 5, i.e., $sensor_{67}$, $sensor_{89}$ and $sensor_{45}$. If $sensor_{12}$ gets replies from all $sensor_{67}$, $sensor_{89}$ and $sensor_{45}$, then the routing advertisement for cost 5 is valid. However, it is not possible to reliably determine if the path having cost 3 is unavailable.

In general, in a hostile environment a sensor can determine *availability* of a path to a subnet but it cannot determine its *unavailability*. By dropping or rerouting a message from the requesting sensor or the replying sensor, a malicious entity can make the requesting sensor believe that a path to a subnet is unavailable. On the other hand, if a malicious entity can drop or re-route packets on a path, the path is unreliable and an unreliable path is as good as not being available at all.

If $sensor_{12}$ finds that the path having cost 3 is available then it is possible that either the routing advertisement of cost 5 is malicious or the routing advertisement is due to a transitory change in the routing configuration. If the routing advertisement is transitory and the path from v_2 to subnet s_{45} at cost 3 is available, then v_2 should eventually advertise a route to subnet s_{45} at cost 3. If $sensor_{12}$ does not see a routing advertisement at cost 3 then the routing advertisement at cost 5 is malicious. If $sensor_{12}$ sees a routing advertisement at cost 3 then the sensor does not verify the path having cost 5 any further.

5.4 Timing Verification

Routers advertise routing messages at certain intervals of time. The interval of time at which RIP messages are advertised is $rip_{interval}$. A router can send more than one RIP message in $rip_{interval}$.[7] $rip^{high}_{threshold}$ is the maximum number of packets that a sensor should ever see within $rip_{interval}$.

A sensor maintains a counter $rip^i_{counter}$ and a timer rip^i_{timer} for each router v_i that is connected to the link on which the sensor is placed. The sensor initializes rip^i_{timer} and sets $rip^i_{counter}$ to 0. It sets the time-out value to $rip_{interval}$. The sensor increments $rip^i_{counter}$ for every RIP advertisement that it sees from router v_i. If $rip^i_{counter}$ is greater then $rip^{high}_{threshold}$ when rip^i_{timer} expires, then this is considered an attack.

The sensor also maintains a value $rip^{low}_{threshold}$, which is the minimum number of packets that a sensor should see within $rip_{interval}$. If $rip^i_{counter}$ is less that $rip^{low}_{threshold}$ when rip^i_{timer} expires, it can be inferred that the router is not working. This could be due to a denial-of-service attack against the router or due to a failure. $rip^{high}_{threshold}$ and $rip^{low}_{threshold}$ are implementation and topology dependent. These values have to be experimentally or statistically determined for a network.

6. EXPERIMENTAL VALIDATION

An experimental testbed network was built to test the vulnerability of routing protocols. The testbed routers are multi-homed PCs, equipped with the *GNU Zebra* routing daemon, version `0.91a`. The testbed contains five different autonomous systems. The autonomous systems exchange routing information using BGP. The interior gateway protocol used within the autonomous systems is either RIP or OSPF. Figure 4 is a schematic of the complete testbed topology. The experiments presented in this paper only use autonomous system 1. The other autonomous systems were used to conduct experiments on OSPF and BGP. Those experiments are not discussed in this paper.

The networks in the testbed have addresses in the range `192.168.x.x`. Each network has a subnet mask of `255.255.-255.0`. In the following, subnet "`x`" is used to denote subnet `192.168.x.0` and address "`x.x`" denotes the IP address `192.168.x.x`.

The following sections discuss three very simple attacks that were carried out on the testbed. These attacks are proof-of-concepts to demonstrate the vulnerabilities in distance-vector routing protocols and how the approach described here is used to detect these attacks.

6.1 Routing Loop Attack

The *Routing Loop Attack* demonstrates how false routing information generated by a remote host can propagate to routers and, as a consequence, install wrong routes in the routing tables.

Consider the network shown in Figure 4. A routing loop for subnet `30` is created by spoofing routing advertisements. Note that subnet `30` does not exist in Figure 4. The spoofed routing advertisements are generated by host *mike*. The source IP address of the spoofed routing advertisements is set to be the address of router *hotel*. The destination of the spoofed routing advertisements is router *foxtrot*. This particular choice of source and destination addresses is dictated by the particular topology of the network. Spoofed routing advertisements with the source address of router *golf* are not forwarded by *golf*. Therefore, the source of the spoofed routing advertisement cannot be *golf*. In addition, routers accept routing information only from their neighbors. Therefore, the source of the spoofed routing advertisements has to be *hotel*.

The spoofed routing advertisements from *mike* are routed through *golf* to reach *foxtrot*. As a consequence, *foxtrot* adds a route for subnet `30` through *hotel*. *foxtrot* then advertises this route to *golf* but not to *hotel*, because it believes that *hotel* is the source of the route. After receiving the advertisement from *foxtrot*, *golf* adds a route to subnet `30`, through *foxtrot*. Then, *golf* sends a routing advertisement for subnet `30` to *hotel*. When the advertisement is processed by *hotel*, a route to subnet `30` through *golf* is added to *hotel*'s routing table. This results in a routing loop. A `traceroute` from host *india* for an address in subnet `30` shows the path *golf-foxtrot-hotel-golf-*$\cdots$*-golf*.

6.2 Faulty Route Attack

The *Faulty Route Attack* demonstrates how a malicious entity can divert traffic by advertising a route with a cost that is lower that the optimal cost. Consider Figure 4. A malicious host on the link between *golf* and *foxtrot* wants to access incoming traffic to subnet `25`. To achieve this, the attacker has to convince *foxtrot* that the best way to reach subnet `25` is to route traffic through *golf* instead of *romeo*. The route associated with *romeo* has cost 2.

Therefore, the malicious host pretends to be *golf* and sends a routing message to *foxtrot* advertising a route to subnet `25` at cost 1. As a consequence, *foxtrot* modifies the route to subnet `25` in its routing table. The new route goes through *golf* instead of *romeo*.

[7] RIP information from one router may be split and advertised in multiple RIP messages.

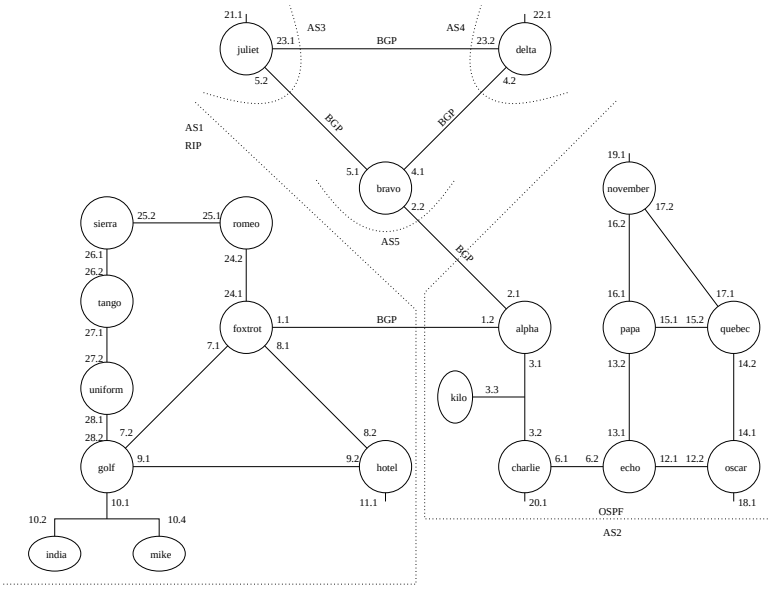

Figure 4: Testbed Topology

6.3 Denial-of-Service Attack

In the *Denial-of-Service Attack*, malicious hosts on links between *foxtrot* and *golf*, *foxtrot* and *hotel* and *foxtrot* and *romeo* collaborate to make subnet `10` unreachable from *foxtrot*. The malicious host on the link between *foxtrot* and *golf* pretends to be *golf* and spoofs routing advertisements to *foxtrot* advertising unreachability to subnet `10`. On receipt of the spoofed routing advertisements, *foxtrot* modifies the route to subnet `10` in its routing table to go through *hotel* instead of *golf*.

Next, the malicious host on the link between *foxtrot* and *hotel* pretends to be *hotel* and spoofs a routing message to *foxtrot* advertising unreachability to subnet `10`. On receipt of the spoofed routing advertisement, *foxtrot* modifies the route to subnet `10` in its routing table to go through *romeo* instead of *hotel*.

When *romeo* too advertises unreachability to subnet `10`, *foxtrot* removes the route to subnet `10` from its routing table because it has no other way to reach subnet `10` except through *golf*, *hotel*, or *romeo*. A `traceroute` from *foxtrot* to subnet `10` returns a `Network Unreachable` error.

The malicious hosts keep sending spoofed routing updates at a rate faster than the actual routing updates, thus preventing the infrastructure from re-establishing correct routes.

These experiments demonstrate how spoofed routing advertisements from unauthorized entities can disrupt the distance-vector routing infrastructure. A malicious entity can advertise non-existent routes, advertise unreachability when a route is actually present, or advertise better-than-optimal routes to divert traffic.

6.4 Detection

A preliminary experimental evaluation of the intrusion detection approach is presented here. A detailed evaluation of the approach is the subject of our current research. The current objective is to establish a proof-of-concept by detecting the attacks outlined in Section 6. Two sensors are placed on the testbed of Figure 4. $sensor_{gf}$ is placed on the link between routers *golf* and *foxtrot*. $sensor_{gm}$ is placed on the link connecting *golf* to subnet `10`. The sensor configurations are generated following the algorithm presented in Section 4.

Consider the *Routing Loop Attack*. Routing advertisements for subnet `30` spoofed by *mike* are analyzed by $sensor_{gm}$ on the link connecting *golf* to subnet `10`. $sensor_{gm}$ detects the source link-level and IP addresses of the routing advertisements to be incorrect. The spoofed routing advertisements have the source link-level and IP addresses of *hotel* whereas the only possible routing advertisement on that link should be from router *golf*. $sensor_{gf}$ also detects the spoofed routing advertisements because subnet `30`, advertised in the spoofed routing message, does not exist in the domain.

Consider the *Faulty Route Attack*. $sensor_{gf}$ detects a routing message that appears to come from router *golf* advertising a route to subnet `25` at cost 1. Therefore, $sensor_{gf}$ searches its configuration for possible paths from *golf* to subnet `25` that do not have *foxtrot* as the first hop. The possible paths are through either router *uniform* or router *hotel* but none of the paths have cost 1. Therefore, $sensor_{gf}$ infers that the advertised cost from *golf* to subnet `25` is an impossible cost. As a consequence, the routing advertisement is considered malicious.

Consider the *Denial-of-Service attack*. When $sensor_{gf}$ receives a spoofed routing message, which appears to be coming from router *golf*, it considers unreachability as a sub-optimal-cost advertisement. Therefore, $sensor_{gf}$ sends a message to $sensor_{gm}$ to validate whether the route to subnet `10` through *golf* is available or not. $sensor_{gf}$ receives a reply from $sensor_{gm}$. As a consequence, $sensor_{gm}$ concludes that the routing message is malicious.

7. ALGORITHM EVALUATION

The experimental evaluation validates the fact that the suggested approach can successfully detect malicious routing advertisements. Nonetheless, our approach has a few limitations, which are discussed here.

7.1 Computational Complexity

The all-pair/all-path algorithm is used to generate sensor configurations. It finds all paths from every router to every other router in the network. This algorithm has an exponential lower bound. However, the algorithm is suitable for medium-size sparse topologies. Several real topologies that run RIP were analyzed. The all-pair/all-path algorithm converged in acceptable time for all these topologies. The algorithm should converge in acceptable time for most topologies running RIP, since these topologies are not very large or dense.

In addition, the sensor configurations are generated offline. The configurations have to be regenerated if routers and links are added to the topology. The sensors have to be brought offline during this time. However router crashes or link failures do not require that the sensor configurations be regenerated.

7.2 Message Overhead

Another drawback of the approach is the additional traffic that is generated to validate optimal and sub-optimal routes. The sensor traffic increases as the number of sensors grows and the required security guarantees become more stringent. Consider a routing advertisement m from router v_i, advertising a path to subnet s_{ab}. If a sensor $sensor_{ij}$ determines that the cost advertised in m is an impossible cost, $sensor_{ij}$ can declare the routing advertisement to be malicious without communicating with any other sensor. Therefore, there is no traffic overhead in this case.

If $sensor_{ij}$ determines that the cost advertised in m is an optimal cost, $sensor_{ij}$ will search its configuration to find all the optimal paths from router v_i to subnet s_{ab}. Let the set of optimal paths from router v_i to subnet s_{ab} be $\{p_1^o, p_2^o, p_3^o, \cdots\}$ and the corresponding set of number of sensors on each path be $\{n_1^o, n_2^o, n_3^o, \cdots\}$. Now, the maximum number of messages that $sensor_{ij}$ will generate to verify m will be $n_1^o + n_2^o + n_3^o + \cdots$. However, if p_1^o is the first optimal path to be verified and p_1^o is found to be a valid path from router v_i to subnet s_{ab}, $sensor_{ij}$ will only generate n_1^o messages to verify m. If p_1^o is the first path to be verified, $n_1^o \leq n_{optimal} \leq n_1^o + n_2^o + n_3^o + \cdots$ where $n_{optimal}$ is the number of messages that $sensor_{ij}$ will generate to verify an optimal-cost advertisement received from router v_i for subnet s_{ab}. Assuming that every sensor replies to every request, the total overhead is $2 * n_{optimal}$ messages.

If $sensor_{ij}$ determines that the cost advertised in m is a sub-optimal cost, $sensor_{ij}$ will search its configuration to find the set of paths $\{p_1^o, p_2^o, \cdots, p_1^{s1}, p_2^{s1}, \cdots, p_1^{s2}, p_2^{s2}, \cdots, p_1^a, p_2^a, \cdots\}$ from router v_i to subnet s_{ab} where p_1^o is an optimal path, p_1^{s1} is a sub-optimal path, p_1^a is a path corresponding to the advertised cost. $\{n_1^o, n_2^o, \cdots, n_1^{s1}, n_2^{s1}, \cdots, n_1^{s2}, n_2^{s2}, \cdots, n_1^a, n_2^a, \cdots\}$ is the corresponding set of number of sensors on each path. Now, the maximum number of messages that $sensor_{ij}$ will generate to verify that all paths from router v_i to subnet s_{ab} with costs less than the advertised cost are not available, is $n_{<advertised} = n_1^o + n_2^o + \cdots + n_1^{s1} + n_2^{s1} + \cdots + n_1^{s2} + n_2^{s2} + \cdots$. If all paths with less than advertised cost are found unavailable, $sensor_{ij}$ will generate $n_{advertised}$ messages to verify that at least one path with the advertised cost is available where $n_1^a \leq n_{advertised} \leq n_1^a + n_2^a + n_3^a + \cdots$. In this case, the total overhead is $2 * n_{<advertised} + 2 * n_{advertised}$.

For a sub-optimal-cost advertisement, $sensor_{ij}$ might find an available path to subnet s_{ab} from router v_i with a cost that is less than the advertised cost. Let the available path be n_2^{s1}. Now, the number of messages generated by $sensor_{ij}$ is $n_{<advertised} = n_1^o + n_2^o + \cdots + n_1^{s1} + n_2^{s1}$. $n_{advertised} = 0$ since no messages are generated to verify paths with the advertised cost unless it is determined that all paths with costs lesser than the advertised cost are unavailable. In this case, the total overhead is $2 * n_{<advertised}$.

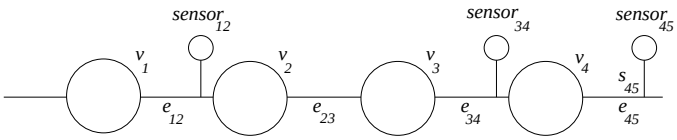

Figure 5: Message Overhead Example

Path verification is done for every routing update that advertises an optimal or sub-optimal cost. Every sensor that detects a routing advertisement will generate an overhead of $2 * n_{optimal}$ for every optimal route and $2 * n_{<advertised} + 2 * n_{advertised}$ for every sub-optimal route in the routing advertisement. However, the overheads $2 * n_{optimal}$ and $2 * n_{<advertised} + 2 * n_{advertised}$ decrease as the proximity of a sensor with the advertised destination increases.

Routing updates are generated every 30 seconds. If we assume the verification packet size to be 48 bytes (including the headers), then every 30 seconds, the verification of one destination will result in an overhead of $48 * 2 * n_{optimal}$ or $48 * 2 * n_{<advertised} + 48 * 2 * n_{advertised}$ bytes depending on whether the advertised cost is optimal or sub-optimal.

For example, consider Figure 5. The cost of all links is equal to 1. When router v_4 advertises a route for subnet s_{45} at cost 1 on link e_{34}, $sensor_{34}$ searches its configuration and determines that an advertisement from v_4 for s_{45} at cost 1 is an optimal-cost advertisement. Since there is only one path $\{(v_4)\}$ from v_4 to s_{45}, and $sensor_{45}$ is the available sensor on that path, $sensor_{34}$ sends a verification request to $sensor_{45}$. $sensor_{45}$ replies with a verification reply to $sensor_{34}$. Since each verification message has a size of 48 bytes, this verification by $sensor_{34}$ requires 96 bytes.

Next, v_3 advertises a route for subnet s_{45} at cost 2 on link e_{23}. Since e_{23} does not have any sensor on it, no verification messages are generated. Next, when v_2 advertises a route for s_{45} at cost 3 on e_{12}, $sensor_{12}$ searches its configuration and determines that an advertisement from v_2 for s_{45} at cost 3 is an optimal-cost advertisement. There is only one path $\{(v_2, v_3, v_4)\}$ from v_2 to s_{45}, and two sensors ($sensor_{34}$ and $sensor_{45}$) are available on that path. Therefore, $sensor_{12}$ sends one message to $sensor_{34}$ and one message to $sensor_{45}$. Both $sensor_{34}$ and $sensor_{45}$ reply back. Therefore, the verification by $sensor_{12}$ requires $48 * 4 = 192$ bytes. The entire verification process requires $96 + 192 = 288$ bytes. This is the amount of overhead that will be generated every 30 seconds.

The overhead due to path-verification might be tolerable under normal circumstances. Under attack conditions, the overhead will be the same as discussed above depending on whether the advertised cost is impossible, optimal or sub-optimal. A malicious entity might try to use the sensors to do a denial-of-service attack by sending an excessive amount of routing updates. However, this will result in the generation of more routing updates than the sensor's threshold. Under such a condition, the sensor will appropriately scale back its path verification mechanism and raise an alert.

However, the overhead due to path-verification may be

unacceptable under conditions where a major part of the network fails. If a link breaks, then every router that is using a path that contains the broken link will send a routing update to its neighbors. Every routing update will generate an overhead as discussed above. Moreover, a link breakage will result in sub-optimal cost advertisements or unreachability advertisements, which are also treated as sub-optimal cost advertisements. The overhead to verify a sub-optimal-cost advertisement is more than that of verifying an optimal-cost advertisement.

If a number of links fail, then routes to many destinations will change or become unavailable, leading to many routing updates. Consequently, there will be an increase in the number of sensor verification messages. When links get reconnected [8], better paths might be advertised and as a consequence, again, increase the number of sensor verification messages.

The message overhead due to the path-verification protocol can be reduced by reducing the number of sensors in the network, reducing the frequency of path verification, or verifying the advertisement selectively. However, all of these approaches will lead to weaker security guarantees. Our present work is focused on reducing the overhead due to path verification without reducing the effectiveness of detection. A possible way of reducing the overhead is by modifying the path verification algorithm so that it does not try to verify every path that corresponds to the advertised cost. Instead of sending a verification message to every sensor on every path having the advertised cost, the verifying sensor sends just one message to an IP address in the destination subnet. This message will traverse a certain path to reach the destination. All sensors present on this path will send back a reply to the verifying sensor. Based on the replies that are sent back, the sensor decides whether there exists a path with the advertised cost that has all the replying sensors on it or not. This approach should reduce the path verification overhead significantly.

7.3 Other Limitations

Our approach is not capable of detecting attacks under certain conditions. Random dropping or modification of data packets and unauthorized access to routers cannot be detected. The verifying sensor uses probe packets to verify that a path with a cost less than the advertised cost is not available. Since, a malicious router present on the path between the verifying sensor and the destination can drop the probe packets, it is not possible to reliably determine that. Consider there is a malicious router on every available path, with a cost less than the advertised cost, from the verifying sensor to the destination. If all these routers drop probe packets, then the verifying sensor can be made to believe that all paths with a cost less than the advertised cost are unavailable. The verifying sensor will then consider a sub-optimal path to be valid even though better paths are present.

Attacks cannot be detected on links where sensors are not present. A malicious entity can advertise false routing information on links where there are no sensors without being detected. The false routing information will be accepted by the routers connected to the link and will be advertised further. The effects of the false routing information will propagate undetected until it reaches a link where a sensor is present.

[8] Again, we consider only those links that were a part of the topology that was used to generate the sensor configurations.

The detection scheme might also generate false positives. Sensors that do not have configurations based on the correct network topology might generate false alarms. Incorrect threshold values and timers also may cause false alarms. For example, when a sensor sends a verification message, it sets a timer within which it expects to receive a reply from the other sensor. If the reply gets delayed due to some network condition, then false alarms may be generated.

False alarms may also be generated when verifying sub-optimal-cost advertisements. To verify a sub-optimal-cost advertisement, a sensor first verifies that all optimal-cost paths are unavailable. If the verifying sensor receives replies from all the sensors associated with an optimal path, it will infer that the optimal path is available. However, if sensors are not placed on every link, there might be a broken link after the last sensor used for the verification of the optimal path. As a consequence, the verifying sensor will assume that the optimal path is available and will generate a false alarm indicating that the sub-optimal path is malicious.

8. CONCLUSION

This paper presented a novel approach to detect attacks against the routing infrastructure. The approach uses a set of sensors that analyze routing traffic in different locations within a network. An algorithm to automatically generate both the detection signatures and the inter-sensor messages needed to verify the state of the routing infrastructure has been devised for the case of the RIP distance-vector routing protocol.

The approach described here has a number of advantages. First, the implementation of our approach does not require any modification to routers and routing protocols. Most current approaches require routers and routing protocols be changed. The high cost of replacing routers and the risk of large-scale service disruption due to possible routing protocol incompatibility has resulted in some inertia in the deployment of these approaches. Our approach, on the other hand, is deployable and provides a preliminary solution to detecting attacks against the routing infrastructure.

Second, the detection process does not use the computational resources of routers. There might be additional load on the routers from having to forward the traffic generated by sensors. However, this additional load should not be as much as it would be if a router had to perform public-key decryption of every routing update that it received, which is what most current schemes require.

Third, the approach supports the automatic generation of intrusion detection signatures, which is a human-intensive and error-prone task.

However, the approach has some drawbacks. First, the complexity of the offline computation that generates sensor configurations increases as the density of the network graph increases. Our experience suggests that this will not be a problem in the case of real-life topologies, but it could become unmanageable for densely interconnected infrastructures. In addition, sensors generate additional traffic to validate routing advertisements. The amount of additional traffic generated increases as the required security guarantees become more stringent. Finally, attacks where subverted routers modify the contents of data packets or drop packets selectively, cannot be detected using this approach.

Our future work will be focused on extending the approach described here to intra-domain link-state protocols (e.g., OSPF) and inter-domain protocols (e.g., BGP). The vulnerabilities associated with these protocols have already

been analyzed in our testbed network and a preliminary version of the algorithm for both the OSPF and BGP protocols has been devised. To address large-scale networks where knowledge of the complete topology cannot be assured, we are working on a model where intelligent decisions can be made based on partial topology information.

Acknowledgments

We want to thank Prof. Richard Kemmerer for his invaluable comments.

This research was supported by the Army Research Office, under agreement DAAD19-01-1-0484 and by the Defense Advanced Research Projects Agency (DARPA) and Rome Laboratory, Air Force Materiel Command, USAF, under agreement number F30602-97-1-0207. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon.

The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Army Research Office, the Defense Advanced Research Projects Agency (DARPA), Rome Laboratory, or the U.S. Government.

9. REFERENCES

[1] S. Axelsson. Intrusion Detection Systems: A Taxomomy and Survey. Technical Report 99-15, Dept. of Computer Engineering, Chalmers University of Technology, Sweden, March 2000.

[2] K.A. Bradley, S. Cheung, N. Puketza, B. Mukherjee, and R.A. Olsson. Detecting Disruptive Routers: A Distributed Network Monitoring Approach. In *Proceedings of the IEEE Symposium on Security and Privacy*, May 1998.

[3] S. Cheung. An Efficient Message Authentication Scheme for Link State Routing. In *13th Annual Computer Security Applications Conference*, December 1997.

[4] S. Cheung and K. Levitt. Protecting Routing Infrastructures from Denial of Service Using Cooperative Intrusion Detection. In *Proceedings of the New Security Paradigms Workshop*, September 1997.

[5] S. Cheung, K.N. Levitt, and C. Ko. Intrusion Detection for Network Infrastructures. In *Proceedings of the 1995 IEEE Symposium on Security and Privacy*, Oakland, CA, May 1995.

[6] M.T. Goodrich. Efficient and Secure Network Routing Algorithms. Provisional patent filing, January 2001.

[7] R. Hauser, A. Przygienda, and G. Tsudik. Reducing the Cost of Security in Link-State Routing. In *Proceedings of the Symposium on Network and Distributed System Security*, February 1997.

[8] L.T. Heberlein, K. Levitt, and B. Mukherjee. An intrusion-detection system for large-scale networks. In *Proceedings of the 15th National Computer Security Conference*, Baltimore, MD, October 1992.

[9] Christian Huitema. *Routing in the Internet*. Prentice Hall PTR, 1995.

[10] Y.F. Jou, F. Gong, C. Sargor, X. Wu, F. Wu, H.C. Chang, and F. Wang. Design and Implementation of a Scalable Intrusion Detection System for the Protection of Network Infrastructure. In *DARPA Information Survivability Conference and Exposition*, January 2000.

[11] S. Kent, C. Lynn, J. Mikkelson, and K. Seo. Secure Border Gateway Protocol (Secure-BGP) - Real World Performance and Deployment Issues. In *Proceedings of the Symposium on Network and Distributed System Security*, February 2000.

[12] S. Kent, C. Lynn, and K. Seo. Secure Border Gateway Protocol (Secure-BGP). *IEEE Journal on Selected Areas in Communications*, 18(4):582–592, April 2000.

[13] G. Malkin. Rip version 2. IETF RFC 2453, Nov 1998.

[14] S.L. Murphy. Presentation on Security Architecture of the Internet Infrastructure. In *Proceedings of the Symposium on Network and Distributed System Security*, April 1995.

[15] S.L. Murphy and M.R. Badger. Digital Signature Protection of the OSPF Routing Protocol. In *Proceedings of the Symposium on Network and Distributed System Security*, February 1996.

[16] R. Perlman. *Network Layer Protocols with Byzantine Robustness*. PhD thesis, Department of EECS, MIT, August 1988.

[17] D. Qu, B.M. Vetter, F. Wang, R. Narayan, F. Wu, F. Jou, F. Gong, and C. Sargor. Statistical Anomaly Detection for Link-State Routing Protocols. In *In Proceedings of the 1998 International Conference on Network Protocols*, October 1998.

[18] Y. Rekhter and T. Li. A border gateway protocol 4 (bgp-4). IETF RFC 1654, Mar 1995.

[19] B.R. Smith and J.J. Garcia-Luna-Aceves. Securing the Border Gateway Routing Protocol. In *Proceedings of Global Internet '96*, London, UK, November 1996.

[20] B.R. Smith, S. Murthy, and J.J. Garcia-Luna-Aceves. Securing Distance-Vector Routing Protocols. In *Proceedings of the Symposium on Network and Distributed System Security*, February 1997.

[21] F. Wang and F. Wu. On the Vulnerablity and Protection of OSPF Routing Protocol. In *IEEE Seventh International Conference on Computer Communications and Networks*, October 1998.

[22] F. Wu, H.C. Chang, F. Jou, F. Wang, F. Gong, C. Sargor, D. Qu, and R. Cleaveland. Jinao: Design and implementation of a scalable intrusion detection system for the ospf routing protocol, February 1999.

[23] F. Wu, F. Wang, B.M. Vetter, W.R. Cleaveland, F. Jou, F. Gong, and C. Sargor. Intrusion Detection for Link-State Routing Protocols, December 1996.

[24] K. Zhang. Efficient Protocols for Signing Routing Messages. In *Proceedings of the Symposium on Network and Distributed System Security*, February 1998.