Small Lies, Lots of Damage: a Partition Attack on Link-State Routing Protocols

Reuven Cohen Raziel Hess-Green Gabi Nakibly
Department of Computer Science
Technion – Israel Institute of Technology
Haifa, Israel

***Abstract*—The Internet consists of a large number of interconnected heterogeneous ASs (Autonomous Systems), each owned and administered by an autonomous organization. Traffic in each AS is forwarded by routers that maintain a coherent picture of the network topology using an intra-AS routing protocol. The most popular intra-AS routing protocols are link-state protocols, such as OSPF and IS-IS. An attacker who compromises a single AS router can send false routing advertisements. In the most simple and practical variant of the attack, the attacker falsifies only its own routing advertisements and not those of other routers. However, such an attack is widely considered to have limited effectiveness, because only a small part of the topology is falsified. In this paper we disprove this conception, by presenting and analyzing a new attack, referred to as a "partition attack," which can cause extensive damage throughout the AS by causing routers to have an incoherent view of the AS topology. We investigate the computational complexity of the attack and show its effectiveness using extensive simulations. An important property of this attack is that it cannot be prevented even if LSAs are digitally signed.**

I. Introduction

The Internet consists of a large number of interconnected heterogeneous ASs (Autonomous Systems), each owned and administered by an autonomous organization. Traffic in each AS is forwarded by routers that maintain a coherent picture of the network topology using an intra-AS routing protocol. The most popular intra-AS routing protocols are link-state protocols, such as OSPF and IS-IS [19]. Link-state routing protocols are also very popular for new network technologies, such as mobile ad-hoc networks [5], sensor networks [2], [20] and Ethernet-based datacenters [6].

Link-state routing protocols work by having each router build a local map of the entire network topology, which it uses to find the shortest path to every destination subnet. To build its own picture of the network, each router creates a link-state advertisement (LSA), which describes its local links and their weights. These LSAs are then disseminated to all the other routers in the AS, and each router uses them to build its local routing map.

Link-state routing protocols have several known security vulnerabilities [1], [10], [19], [24], [25]. The most important are related to the creation and dissemination of incorrect LSAs by a malicious router residing within the AS. There are two variants of such an attack:

This research was partially funded by a grant from the Israeli Ministry of Science and Technology.

1) Self-LSA falsification, where the malicious router falsifies only its own LSA.
2) Other-LSA falsification, where the malicious router advertises a false LSA on behalf of other routers within its AS.

Since every LSA describes only a small portion of the AS topology, that is, a single router and the links to its immediate neighbors, a self-LSA falsification attack is thought to have only a limited effect on each router's view of the AS topology. In contrast, other-LSA falsification attack can "poison" a large piece of the AS topology viewed by all the routers, and are therefore believed to cause more extensive damage. However, other-LSA falsification has two major caveats, which self-LSA falsification does not have:

1) When a router receives a false LSA that was advertized by another router on its behalf, it immediately triggers what is known as the "fight-back" mechanism [22], [25]. This mechanism advertises a newer instance of the LSA, which cancels any impact of the false one. A few attacks have been proposed to circumvent the "fight-back" mechanism [15], [17], [9], but all of them are either complex, or are not effective for every AS topology or router vendor. Therefore, in the general case, an attacker who wishes to persistently falsify LSAs of other routers must repeatedly send out the false LSAs, thus increasing the chances of its exposure.
2) Defense mechanisms that employ a digital LSA signature [13], [18] completely mitigate the possibility this attack, even if the fight-back mechanism is circumvented.

As noted above, since self-LSA falsification can have a limited effect on other routers' view of the AS topology, it is widely believed [14], [15], [17], [16] that such an attack has a limited power. For example, if its goal is to overload links and routers in the AS, it is believed that an attacker can do that only in its vicinity, by falsely advertising that it is directly connected to many destinations. This will cause a lot of traffic to these destinations to be routed towards the attacker's vicinity. This attack is not only limited in its scope, but also can be easily exposed using dataplane probes, such as tarceroute. This paper shows that the network security community has a serious misconception, because self-LSA falsification attacks have a much greater power compared to what is widely believed. In

particular, we show that a judicious use of such attack can overload remote links and routers that are far from the attacker, while leaving no obvious dataplane traces that may lead to the attacker.

We show this by proposing and analyzing a special case of the self-LSA falsification attack, referred to as a "partition attack." We investigate the effects of such an attack using both computational complexity analysis and extensive simulations. The essence of this attack is that it not only falsifies the view other routers have of the AS, but also prevents them from having a joint synchronized view. Consequently, forwarding loops are created, routing paths are lengthened, and routers become disconnected. More importantly, since this is a self-LSA falsification, this attack cannot be prevented even if LSAs are digitally signed.

The main idea behind the new attack is that the attacker partitions the network into two parts, and "convinces" each part to build a different routing map. This is accomplished by having a compromised router send different LSAs on different outgoing ports. These different LSAs have the same header (sequence number, checksum and age fields) , and are therefore considered identical by every router that receives them [15].

Figure 1(a) shows an example of the new attack. Suppose that the compromised router R_0 wants to create routing loops in the network. To this end, it sends different LSAs to R_1 and R_2. In LSA_1, which it sends to R_1, R_0 does not indicate the existence of the link $R_0 - R_1$. In LSA_2, which it sends to R_2, R_0 does not indicate the existence of the link $R_0 - R_2$. Consequently, R_1 and R_2 build different network pictures, as shown in Figure 1(b) and Figure 1(c) respectively. In addition, LSA_1 and LSA_2 are disseminated to the other routers. Routers that receive LSA_1 before LSA_2 build the same network picture as R_1 does, while the other routers build the same network picture as R_2 does. For the sake of our example, suppose that R_1, R_3, R_4, R_9 and R_{10} receive LSA_1 first, while the other routers receive LSA_2 first. If the weights of all links are equal, a routing loop will be created between R_6 and R_3. Consequently, when a data packet is sent from R_8 to R_0, R_8 forwards it to R_3 through R_6, but R_3 forwards it back to R_6.

The main contribution of this paper is to show that self-LSA falsification attacks, although simple and limited in nature, can cause global damage to the entire AS and not only to the vicinity of the attacker. Thus, such an attack should be regarded as a serious threat. Therefore, defenses that rely only on digital signatures (which are ineffective against self-LSA falsification) are insufficient for link-state routing protocols.

The rest of this paper is organized as follows. In Section II we present related work on link-state routing attacks. In Section III we discuss the "partition attack" in greater detail and compare it to previously known attacks. In Section IV we analyze the attack in order to discover how an attacker can attain maximum damage. In Section V we discuss detection and mitigation strategies. In Section VI we present simulations on real network topologies, and in Section VII we conclude the paper.

II. Related Work

Many attacks have been devised for link-state routing protocols that falsify routing LSAs. A powerful variant of this attack is the other-LSA falsification attack, in which a compromised router sends a false LSA on behalf of another router, in order to poison the LSA database of the other routers in the network. An example for this attack is given in [4], where the compromised router sends an LSA on behalf of another router R. This LSA has a higher sequence number than the last LSA sent by R. If R receives its own LSA with incorrect data (i.e., different than the data it sent in its most recent LSA), it invokes the fight-back mechanism [22], [23]. That is, it sends a new (correct) LSA with a higher sequence number. One variant of this false LSA attack is shown in [23] to have a lasting impact if the false LSA is broadcast by the attacker again and again at specific times, each time with a new sequence number. As noted earlier, these attacks can be prevented if LSAs are authenticated using standard cryptographic schemes as presented in [3], [13]. However, such schemes are not the common practice today.

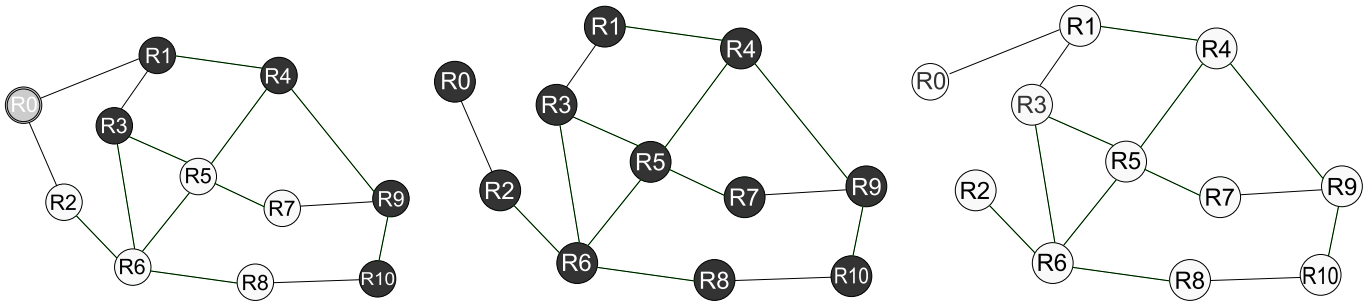
A simpler and more practical attack variant is self-LSA falsification, in which a compromised router produces and sends its own LSAs with false information [22], [24], [25]. This attack is more powerful than the previous one in the sense that it cannot be stopped by invoking the fight-back mechanism and it cannot be prevented using cryptographic schemes.

Two attacks that enable a compromised router to change the LSAs of another router are proposed in [15], [14] for OSPFv2. These attacks can be prevented if the LSAs are digitally signed and authenticated. But if the LSAs are not authenticated, these attacks cannot be stopped by invoking the fight-back scheme. In the first attack, a special feature of OSPFv2 is used to convince a victim router that it can create a link to a non-existing router. This link can then be used to harm the routing in the network in a few ways. The second attack proposed in [15], [14], called "disguised LSA," exploits the fact that LSAs in OSPF are not digitally signed. This allows a compromised router to send LSAs on behalf of another router R without triggering the fight-back mechanism.

A simple attack proposed in [8] is for a compromised router to forward packets not on its shortest paths. In this way, the attacker can create forwarding loops without tampering with the routing protocol. This attack is limited in its ability to create forwarding loops, because each such a loop must contain a neighbor of the compromised router and the shortest path to it from that neighbor.

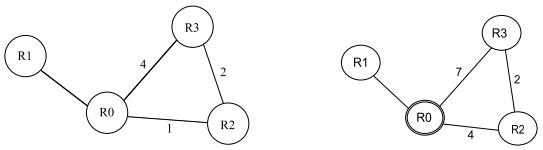
III. The Partition Attack and Its Strength Compared to Previous Attacks

In this section we show that a self-LSA falsification attack can cause a significant damage to the AS, well beyond the attacker's immediate neighborhood. To show this, we devise a new attack type, to which we refer as "partition attack". In the new attack the compromised router sends different LSAs to different neighbors. Both LSAs pretend to represent the correct link states of the compromised router. The different LSAs have



(a) network topology (b) network picture at R_1 and the black nodes (c) network picture at R_2 and the white nodes

Fig. 1. An example of the proposed attack



(a) (b)

Fig. 2. Demonstrating the various attacks

the same header (sequence number, checksum and age fields), but they report different information[1]. While one of these LSAs might be correct, at least one of them contains incorrect information. LSA authentication and fight-back cannot prevent or stop this attack because the compromised router does not change the LSA of any other router. An interesting case of this attack is where the compromised router R sends two different LSAs to two neighbors R_1 and R_2. In the LSA it sends to R_1, R does not report about its link to R_1, and in the LSA it sends to R_2, it does not report about its link to R_2. We call this a "partition attack".

At first glance it seems that this attack has a similar effect to the one where the attacker sends the same incorrect LSA to all its neighbors. However, as shown later, our new partition attack is much stronger. To discuss its impact, we now compare it to three known families of attacks.

Attack-1: In this attack, the compromised router changes its own routing table. Consequently, traffic might be diverted from its shortest paths, and forwarding loops might be created. Each forwarding loop includes the shortest path from the compromised router to one of its neighbors. For example, consider Figure 2(a) and let R_0 be the compromised router. Suppose that when R_0 receives packets for R_1, it sends them to R_3 rather than to R_1. Consequently, these packets will be stuck in the loop $R_0 - R_3 - R_2 - R_0$.

[1]In order to pass the checksum check made by OSPFv2 routers when they decide whether two LSAs are indeed identical, identities of dummy routers can be added at the end of the LSA as needed [15]. These adjacencies do not affect the routing protocol, because OSPF uses only bidirectional links.

Attack-2: In this attack, the compromised router sends a false LSA on behalf of another router, assuming that LSAs are not digitally signed and that the fight-back mechanism can be thwarted (as proposed in [15]). For example, consider again Figure 2(a) and let R_0 be the compromised router. Suppose that $R0$ sends a false LSA on behalf of R_2, and indicates in this LSA that the link $R_2 - R_0$ has a weight higher than 3, or that the link is down. Suppose also that R_3 accepts this LSA and adds it to its database. Consequently, longer paths will be chosen for the traffic sent from R_3 to R_0 and R_1. This attack can also create forwarding loops.

Attack-3: In this attack, the compromised router sends its own LSA with false information. For example, consider again Figure 2(a), and let R_2 be the compromised router. Suppose that the LSA sent by R_2 reports that the cost of the link $R_2 - R_0$ is 3 (or higher) rather than 1. This will cause R_3 to send packets to R_0 over the link $R_3 - R_0$ rather than over the shortest path $R_3 - R_2 - R_0$.

We now distinguish between three possible consequences of the above attacks:

Cons-1: loops that do not necessarily include the attacker are created.

Cons-2: only one loop is created, and this loop includes the attacker.

Cons-3: traffic is diverted from its default (shortest) path to another path, but forwarding loops cannot be created.

Generally, sub-optimal routes are less dangerous than forwarding loops, because traffic eventually reaches its destination and fewer resources are consumed. Thus, Cons-1 and Cons-2 cause greater damage than Cons-3 and Cons-1 clearly does more damage than Cons-2 because it is more general, and the attacker is less likely to be detected.

The new partition attack can create forwarding loops that do not include the compromised router, as shown in Figure I. Thus, both Cons-1 and Cons-2 are possible. Cons-3 is also possible in the new partition attack for the same reason it is possible following Attack-3. In fact, Attack-3, can be viewed as a private case of our partition attack, where the compromised router sends the same false LSA to every neighbor.

attack	without digital signatures	with digital signatures
new attack	Cons-1 Cons-2 Cons-3	Cons-1 Cons-2 Cons-3
Attack-1	Cons-2 Cons-3	Cons-2 Cons-3
Attack-2	Cons-1 Cons-2 Cons-3 are possible but unlikely and uncontrollable	None
Attack-3	Cons-3	Cons-3

TABLE I
THE NEW PARTITION ATTACK IS MORE POWERFUL THAN ANY PREVIOUSLY KNOWN ATTACK

Attack-1 cannot bring about Cons-1 because only one network picture exists, and all the routers except the compromised one follow this picture. A consistent network picture means that the same shortest paths are calculated by all routers. Thus, Attack-1 can only create forwarding loops that contain the compromised router (Cons-2), as illustrated in the demonstration of this attack in Figure 2(a). Cons-3 is also possible due to this attack, because the compromised router can divert traffic from a shortest path without creating forwarding loops.

Attack-2 can theoretically bring about all three consequences, since it may cause different routers to have different network pictures: some use the attacker's LSA, and the others use the correct LSA. However, these consequences are only a side-effect of this attack, when the attack fails to disseminate the false LSA to all the routers. In particular, unlike in our new attack, an attacker that invokes Attack-2 cannot control which part of the network will receive which LSA. An example of two different routers obtaining a different picture of the network topology due to Attack-2 is illustrated in Figure 2(b). Suppose that the compromised router R_0 sends a false LSA on behalf of R_3. This LSA says that the cost of link R_3-R_0 is 1 rather than 7. Now, suppose that router R_2 uses the fake LSA, while R_3 has the correct network picture. Consequently, R_2 will send packets for R_1 through R_3, but R_3 will forward these packets through R_2, which results in a forwarding loop. The same scenario is possible if R_1 or R_2 are the attackers.

Attack-3 leaves the network consistent, and therefore no forwarding loops are possible. Still, Cons-3 is possible if the compromised router says that the metrics of its local links are better (smaller) than their actual values. Consequently, other routers might forward packets to this router rather than on the correct shortest paths. If the compromised router says that the metrics of its local links are worse than their real values, other routers might use longer paths.

Thus, the proposed new attack and Attack-2 inflict greater damage than all other link-state routing attacks. However, the new partition attack is much stronger even than Attack-2, because it cannot be prevented even if cryptographic authentication schemes, based on digital signatures, are used [12], [13]. Such schemes allow every router to verify that an LSA disseminated by a certain router was indeed created by this router and was not modified on its way. Thus, digital signatures prevent Attack-2, but cannot prevent our new attack.

Table I summarizes the above discussion. In this table we indicate the possible consequences of each attack with and without digital signatures.

IV. MAXIMIZING THE IMPACT OF THE PARTITION ATTACK

In this section we address the question how to maximize the impact of the new partition attack on a network. We start with a general theoretical model, where the attacker has no knowledge about the propagation delays of the LSAs. We show that finding the most powerful attack in such a case is computationally intractable. Then, we consider a more practical case where the propagation delays of the LSAs are predictable, and find the most powerful attack in this case.

A. Unpredictable Propagation Delays

We start with a few definitions:

Definition 1: A router R_1 is said to be a part of a forwarding loop with respect to a given destination if it forwards a packet to one of its neighbors, but receives the same packet later on.

As described in Section III, a compromised router that sends a single false version of its own LSA (Attack-3) cannot create a forwarding loop. However, by sending different LSAs to different neighbors, an attacker who invokes the new attack can partition the network into two sets of routers, each with a different network picture. This allows forwarding loops to be created. We define a partition attack as follows:

Definition 2: A partition attack $P_A = (R_a, V_1, V_2, LSA_1, LSA_2)$ is an attack that partitions the network graph $G(V,E)$ into two subgraphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$, where $V_1 \cup V_2 \cup \{R_a\} = V$, $V_1 \cap V_2 = \emptyset$, and R_a is connected to at least one node in V_1 and one node in V_2. The attacking node sends two different LSAs with the same header (sequence number, checksum and age fields): LSA_1 and LSA_2. The nodes of V_1 receive LSA_1 before LSA_2 and the nodes of V_2 receive LSA_2 before LSA_1.

To assess the potential damage of an attack, we focus on the number of source-destination pairs whose traffic is caught in a forwarding loop. Other possible measures could be the number of source-destination pairs whose traffic is directed away from their shortest paths, or the number of such pairs whose traffic is directed through routers that participate in a routing loop. We choose the first measure because routing loops can rapidly collapse the entire network, and because there is a strong correlation between this measure and the latter two.

Definition 3: The damage of a partition attack P_A, denoted $D(P_A)$, is the number of source-destination pairs whose packets are caught in a forwarding loop as a result of this attack.

We present Algorithm 1, which finds the damage inflicted by a given attack. Essentially, the algorithm traverses the routing path for each source-destination pair (s,d) in a search for a forwarding loop. To accomplish this, two spanning trees are created: SPT_1 and SPT_2. The first is a collection of shortest paths to d according to LSA_1 and the second is the same, but according to LSA_2. For each router on the routing path, the next-hop router is determined from the spanning tree induced by the LSA this router holds.

The algorithm is invoked for each source-destination pair, namely $\binom{|V|}{2} = O(|V|^2)$ times. Traversing the path for each pair takes at most $|V|$ steps. Thus, the total time complexity

Algorithm 1: Determining the damage of an attack

Input: A network graph $G = (V, E)$ and an attack: $(R_a, V_1, V_2, LSA_1, LSA_2)$
Output: The damage of the attack
$damage \leftarrow 0$
for *each source-destination pair* (s, d) **do**
  (1) calculate two shortest path trees to d: SPT_1 and SPT_2, according to LSA_1 and LSA_2 respectively; **(2)** start with s, and check what is the next hop to d according to SPT_1 or SPT_2, depending whether s holds LSA_1 or LSA_2 respectively;
  (3) repeat this process for the next hop until d is reached or until a loop is detected.
  (4) if a loop is detected then $damage \leftarrow damage + 1$
return *(damage)*

without calculating SPT_1 and SPT_2 is $O(|V|^3)$. The shortest path trees SPT_1 and SPT_2 can be computed only once per node s, and thus takes $|V|(|E|+|V|log|V|)=O(|V^3|)$. Thus the total time complexity of the algorithm is $O(|V|^3)$.

Algorithm 1 is only valuable when we know which node receives each LSA. However, the main questions faced by the attacker are which of its neighbors to include in each LSA, what is the most effective partition of the graph to V_1 and V_2, and how to ensure that the nodes in V_1 indeed receive LSA_1 before LSA_2 and vice versa. To address these questions, we present the following lemmas:

Lemma 1: Every forwarding loop created by a partition attack $P_A = (R_a, V_1, V_2, LSA_1, LSA_2)$ contains at least two edges between V_1 and V_2, or exactly one edge if the loop consists of only two nodes.

Proof: If no such a link exists, all the routers on the loop belong to the same subgraph. This contradicts the fact that all the routers in the same subgraph have the same network picture, and should calculate the same shortest paths. If such a link exists, there must be at least two such links in a loop. ■

The next lemma characterizes the links that might become part of a forwarding loop as a result of a partition attack.

Lemma 2: Let C be any simple cycle in the network graph, and let $R_a \in C$ be a compromised router. There exists an attack by R_a, which creates a forwarding loop that includes at least one router R, where $R \neq R_a$ and $R \in C$.

For example, consider Figure 3(a) where the weight of every link is 1. Due to an attack by R_a, routers R_1, R_3, R_4 and R_5 have a network picture as in Figure 3(b), whereas R_2 and R_6 have a network picture as in Figure 3(c). Consequently, a forwarding loop is created over the link $R_3 - R_6$, because router R_6 calculates a shortest path towards R_a via R_3, and R_3 calculates a shortest path towards R_a via R_6. This link is part of the following simple cycles: (a) $R_a \rightarrow R_1 \rightarrow R_3 \rightarrow R_6 \rightarrow R_2 \rightarrow R_a$; (b) $R_a \rightarrow R_1 \rightarrow R_3 \rightarrow R_5 \rightarrow R_6 \rightarrow R_2 \rightarrow R_a$; and (c) $R_a \rightarrow R_1 \rightarrow R_4 \rightarrow R_5 \rightarrow R_3 \rightarrow R_6 \rightarrow R_2 \rightarrow R_a$.

Proof: Suppose that R_1 and R_2 are the neighbors of the compromised router R_a, and they sit on a simple cycle C. If R_a has other neighbors, they are omitted from the LSAs it sends, and thus are not considered as neighbors of R_a. We build an attack that creates a forwarding loop that contains R_1 using the following partition: $V_1 = \{R_1\}$, $V_2 = V - \{R_a, R_1\}$. LSA_1, sent by R_a to the routers in V_1, does not include the link $R_a - R_1$, and LSA_2, sent by R_a to the routers in V_2, does not include the link $R_a - R_2$. Thus, the shortest path to R_a calculated by R_1 is via a router in V_2, while the shortest path to R_a calculated by every router in V_2 is via R_1. Thus, a forwarding loop containing R_1 and a router from V_2 is established. ■

From Lemma 2 we learn that the new partition attack can create forwarding loops in many parts of the network. For another example, consider again Figure 3(a), but this time with $V_1 = \{R_1, R_3\}$. In this case, a forwarding loop is created over the link $R_3 - R_1$ as opposed to the one created over the link $R_3 - R_6$ in the previous example. From Lemma 2 we also see a correlation between the damage inflicted by an attack and the number of simple cycles the compromised router participates in: more network cycles create more options for creating forwarding loops.

In order to find a partition with the largest number of potential forwarding loops, we now define a new optimization problem, called *Potential Loops Per Attacker (PLPA)*. Formally, given an undirected graph $G = (V, E)$ and a compromised router R_a with at least two neighbors, the goal is to find a partition of V into two disjoint sets V_1 and V_2 such that (a) each subset is connected; (b) $R_a \notin V_1$ and $R_a \notin V_2$; (c) R_a has an edge to a node in V_1 and to a node in V_2; (d) the number of edges between nodes in V_1 and nodes in V_2 is maximized. By Lemma 1, each loop must contain at least one edge from the cut. Thus, by maximizing the size of the cut, we maximize the number of potential loops.

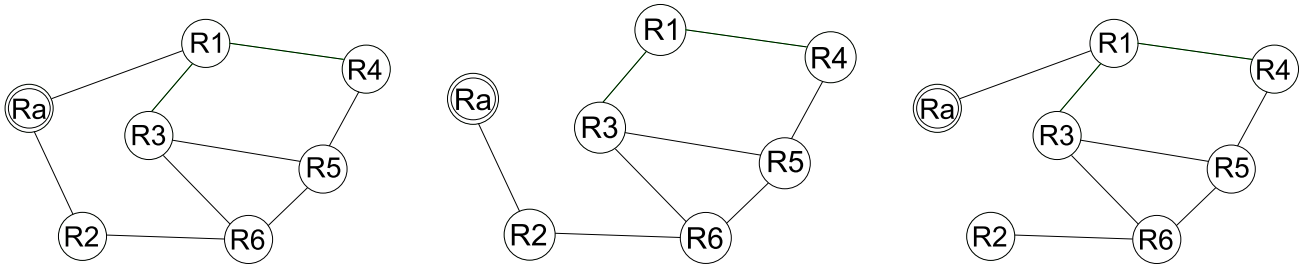
Lemma 3: Finding the number of potential[2] forwarding loops that can be created by a compromised router R_a is NP-complete.

Proof: CMC (Connected Maximum Cut) is a variant of the maximum cut problem, and it is defined as follows. Given an undirected weighted graph $G = (V, E)$, the objective is to find a partition of V into two disjoint subsets V_1 and V_2, such that each subset is connected and the sum of the weights of edges in the cut between V_1 and V_2 is maximum. It is easy to see that there is a reduction from the CMC problem, which is NP-complete (even in planar graphs) [7], to PLPA. ■

B. Predictable Propagation Delays

We now consider the case where the propagation delays and processing times of the LSAs are roughly known to the attacker. In other words, the attacker is able to predict which nodes will be in V_1 and which in V_2. For simplicity, we assume that the propagation delay and processing time is one time unit for every hop. However, a similar algorithm can be used when these times are different for different hops.

[2]Not every potential loop becomes a real loop.



(a) network topology (b) network picture at R_1, R_3, R_4 and R_5 (c) network picture at R_2 and R_6

Fig. 3. An example of Lemma 2 (R_a is the attacker)

To find the attack with maximum impact for a given attacker R_a, we consider every pair x and y of R_a's neighbors. Then, we assume that LSA_1 reports the existence of all the neighbors of R_a except x, whereas LSA_2 reports the existence of all the neighbors of R_a except y. We further assume that LSA_1 is sent to x and LSA_2 is sent to y exactly $[t + \frac{1}{2}]$ time units later. The "$\frac{1}{2}$" is needed to avoid race conditions; it can be replaced with any $0 < \epsilon < 1$, but $\epsilon = \frac{1}{2}$ is the safer value. To find the optimal value of t, we need to consider only $t = 0, 1, ...D$, where D is the network diameter, because if we wait longer than D, all the nodes will receive LSA_1, while LSA_2 will have no effect. Algorithm 2 presents the pseudo-code of the above procedure. Note that the external "for" loop is performed twice for each pair of routers R_1 and R_2: once for $v_1 = R_1$ and $v_2 = R_2$, and once for $v_1 = R_2$ and $v_2 = R_1$.

Algorithm 2: Finding the maximum damage attack, when propagation times are known

Input: A finite graph $G = (V, E)$, and an attacker R_a
Output: The attack by P_A with maximum damage on G, and its value
for ***each*** *two neighbors of* R_a, v_1 *and* v_2 **do**
  for $t = 1$ ***to*** *the diameter of* G **do**
    (1) for ***each*** *node* $v \in G$ **do**
      if the distance from R_a to v through v_1 is longer than t + the distance from R_a to v via v_2 then add v to V_1, else add v to V_2
    (2) $LSA_1 \leftarrow$ all neighbors of R_a except v_1;
    (3) $LSA_2 \leftarrow$ all neighbors of R_a except v_2;
    (4) invoke Algorithm 1 to compute the damage for $P_A = (R_a, V_1, V_2, LSA_1, LSA_2)$;
    (5) save P_A and its damage if maximal
return $(P_A, damage)$

The time complexity of the algorithm consists of the 3 "for" loops and the execution of Algorithm 1. The first loop goes over all pairs of R_a's neighbors, and therefore does not take more than $\binom{deg}{2} = O(deg^2)$ time, where deg is the degree of the network. The second is invoked D times, where D is the network diameter. The third loop runs $O(|V|)$ times, but each time it invokes Algorithm 1, whose time complexity is $O(|V|^3)$. Therefore, the total time complexity is $O(D \cdot deg^2 \cdot |V|^3)$.

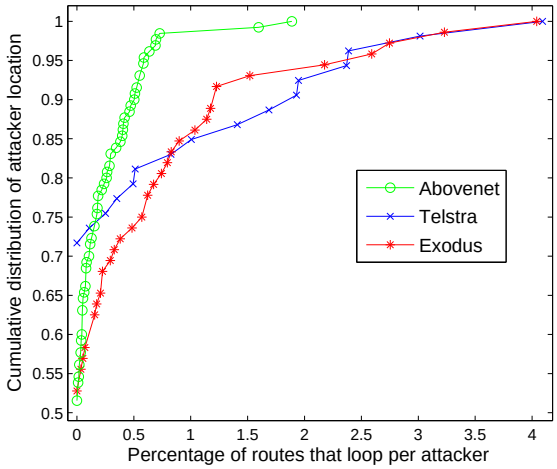
V. Detection and Mitigation

After demonstrating the partition attack and its potential impact on the network, we study in this section possible detection and mitigation strategies. To preserve the distributed nature of the underlying routing protocol, we focus on solutions that do not need the help of a centralized entity.

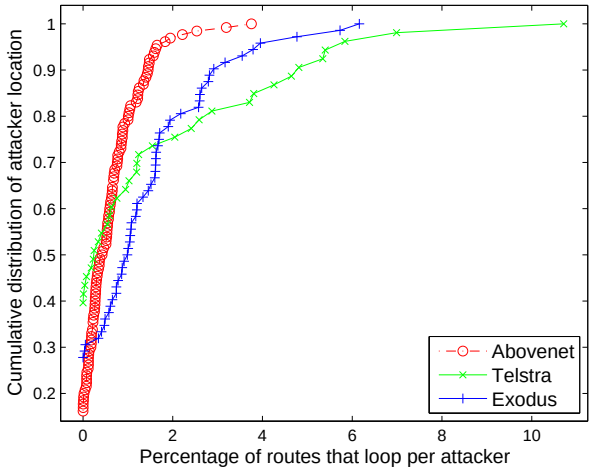
In order to improve the security of link-state routing protocols, several cryptographic measures have been proposed. For instance, [3], [11] propose hash authentication mechanisms, and [13], [18] propose to add digital signatures to the LSAs. These mechanisms prevent one node from distributing false LSAs on behalf of other nodes. However, digital signatures do not prevent the partition attack because they can only be used for LSA authentication. If an attacker distributes two different LSAs with its correct signatures, both LSAs will be accepted and treated as legitimate.

Recall that in a link-state routing protocol, every router receives each LSA multiple times (once from each neighbor). The first copy is accepted and every other copy is ignored. A straightforward way to detect the new attack is for each router to compare the content of any non-first LSA it receives to the content of the first (accepted) LSA with the same sequence number. The LSAs with the same sequence number but different content may be indicative of an attack.

While the above approach will enable some of the routers to detect a compromised router, it requires changing the specifications of the standard link-state routing protocols. In addition, it requires the routers to spend many CPU cycles on the detection of this attack. To facilitate the comparison of the LSAs, a cryptographic hash function can be used to create a digest for each LSA. Then, each router needs only verify that all the LSAs it receives from a given source and with the same header have the same digest. Another practical variant of this defense is that for two LSAs that are considered identical, according to the ordinary rules of the protocol, the one with a greater numerical hash value is considered newer. This ensures that only one LSA will be installed throughout the network, which



(a) equal propagation times of LSAs



(b) varied propagation times of LSAs

Fig. 4. The cumulative damage inflicted by each router using the new attack

reduces the impact of the new attack and makes it similar to that of Attack-3 (see Table I).

The core vulnerability that makes our partition attack possible is that different routers consider different LSAs as identical even though their content is different. Thus, another way to detect the new attack is for every two neighboring routers to compare the first (accepted) LSA they receive for any originating router and for every sequence number. This approach works because, by Lemma 1, a successful attack requires two neighbors whose first (accepted) LSAs are different.

Both of the above approaches can detect any case where different LSAs are sent to different neighbors. However, the specific partition attack discussed in this paper has a unique property: the attacker R_a sends an LSA to a neighbor R, but the LSA does not report that the link $R_a - R$ exists. Thus, this specific attack can be prevented if a router R_2 accepts an LSA originated by its neighbor R_1 and discovers that the link $R_1 - R_2$ is not reported in this LSA. However, this simple approach is not compliant with standard link-state routing protocols, such as OSPF and IS-IS: these protocols make a clear distinction between establishing adjacencies and LSA dissemination; thus, there are valid cases where a router does not advertise a link to a neighbor.

VI. Simulation Study

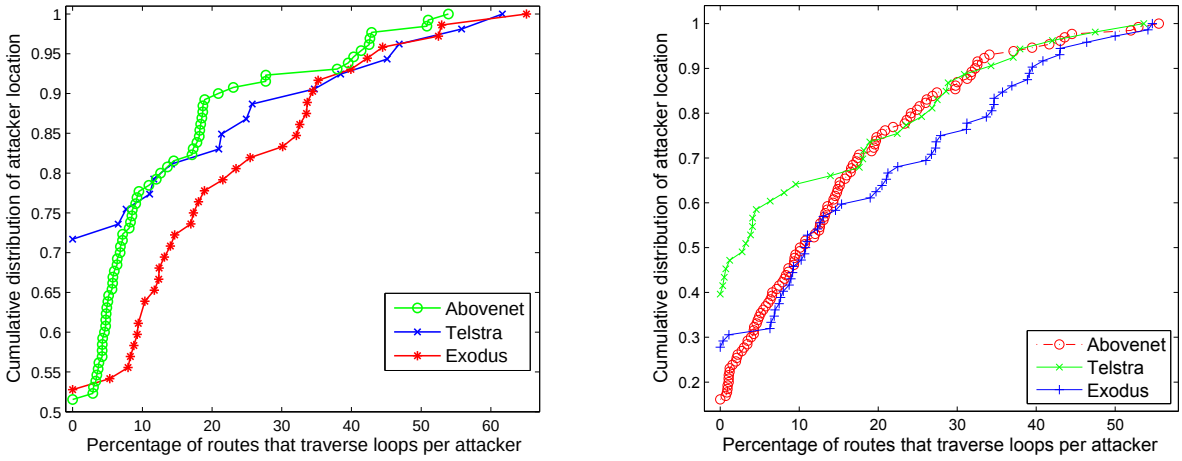
In this section we evaluate the impact of the new partition attack on real network topologies, from the RocketFuel project [21]. RocketFuel is an open ISP topology mapping engine, which can be used to build AS topologies. We invoked the new attack on each of the three AS topologies described in [21], whose details are given in Table II.

ISP name	AS number	number of routers	average degree
Telstra	1221	115	1.3
Exodus	3967	80	1.8
Abovenet	6461	145	2.6

TABLE II
THE THREE TOPOLOGIES USED FOR EVALUATING THE NEW ATTACK

For each AS topology, we invoke Algorithm 2 from every node that has at least two neighbors. Our aim is to find the attack with maximum damage, i.e., the one that maximizes the number of source-destination pairs whose traffic is caught in a forwarding loop, assuming that LSA propagation delays and processing times are predictable.

Figure 4 shows the maximum damage for each tested topology and for every attacker node in the network. In Figure 4(a) we examine the case where propagation delays and processing times of the LSAs are equal on all links. This case is interesting because an attacker can use it to easily predict which routers should be compromised in order to generate a successful attack. In this graph, we first find, using Algorithm 2, the attack that does the most damage when invoked from each router. Then, we calculate the cumulative distribution of the percentage of source-destination pairs whose traffic is caught in a forwarding loop for all three AS topologies. The y-axis indicates the cumulative distribution of attackers whose attack caused the traffic between a specific x-value percentage to be caught in forwarding loops. We assume that traffic is uniformly distributed between all router pairs. For example, consider the line corresponding to Exodus in Figure 4(a). The point $[x = 1, y = 0.84]$ indicates that 16% of the routers are able to produce an attack that catches at least 1% of the traffic in a forwarding loop.



(a) equal propagation times of LSAs

(b) varied propagation times of LSAs

Fig. 5. Cumulative distribution of the percentage of disturbed routing paths per attacker location

While Figure 4(a) examines the specific case of constant processing and propagation delays, Figure 4(b) examines the average of 60 cases with variable delays, chosen according to a random, exponentially distributed, probability. The x and y axes are similar to those in Figure 4(a). In this case we still assume that the delays are known to the attacker

From both graphs we learn that the attack is slightly less effective in the denser topology (Abovenet). In the two denser topologies, however, low-damage attacks can be generated more easily than in the sparse topology (Telstra). A possible explanation is that the elimination of a single link from the LSA of the attacking router is less crucial in a dense topology than in a sparse one. Thus, high-damage attacks are harder to create in a dense topology. But, the existence of a variety of routes in a dense topology makes it easier to generate low-damage attacks.

While the proposed attack mainly affects packets caught in forwarding loops, packets that traverse through a loop of other packets are also affected, because such a loop is likely to be heavily loaded. In Figure 5 we show the cumulative percentage of source-destination pairs whose packets are affected in this way for equal and varied delays. The x and y axes are similar to those in Figure 4. In this graph we ignore source-destination pairs whose packets enter a loop. Thus, the actual percent of source-destination pairs that are affected by an attack is the sum of the percentage in Figure 4 and in Figure 5.

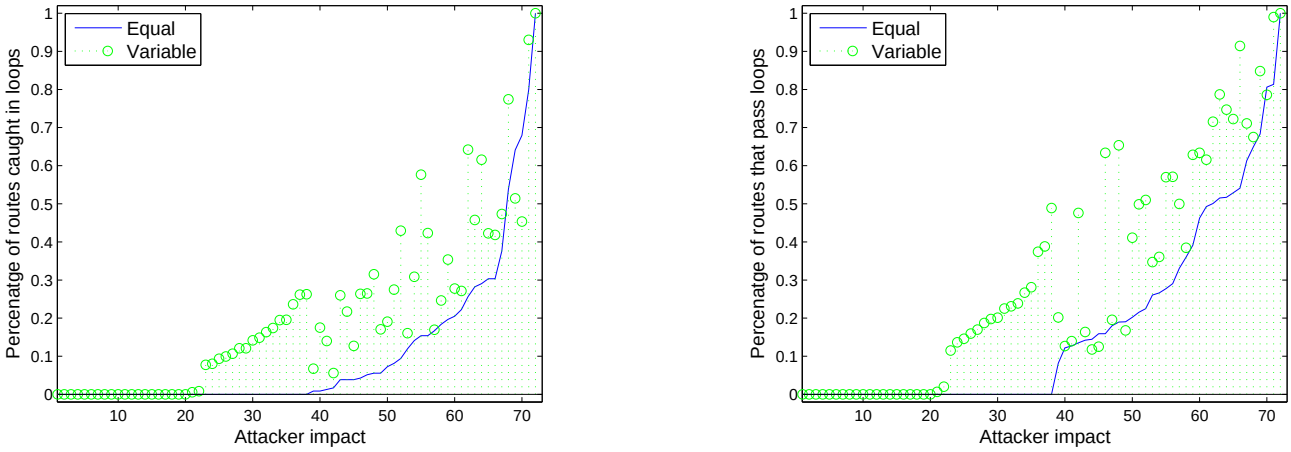
Comparing Figure 4 to Figure 5 reveals that although only a small percent of the routers can generate a high damage attack, about 30% are able to generate an attack that affects more than 15% of the network traffic, in all three topologies. In the sparse topology, about 40% of the routers can generate an attack that affects about 10% of the traffic, whereas in the two denser topologies about 70% of the routers can generate such an attack.

In Figure 6 we study the correlation between the damage when delays are equal and when they are variable. In Figure 6(a), the x-axis indicates the attackers (routers) sorted by the damage generated by their attacks when delays are fixed. The y-axis indicates the damage of the attacks when delays are fixed and when they are variable, in the solid line and in the circles respectively. Each case is separately normalized.

For example, consider the point $[x = 60, y = 0.2]$ on the solid line, and the circle at $[x = 60, y = 0.25]$. The x-value indicates that there is a router whose attack is more successful than 59 of the 72 routers in the equal propagation case. The solid line y-value (0.2) indicates that the damage of this router's attack is 20% of the maximum damage reached when delays are fixed. The circle y-value (0.25) indicates that when delays are variable, the same router is able to generate, on the average, damage of about 25% of the maximum when delays are variable. Figure 6(b) is the same as Figure 6(a), but this time with respect to the number of affected source-destination pairs whose packets are affected by loops of other flows.

We see a very good correlation between the fixed delay and the variable delays. Namely, a successful attacker in the case of fixed delays is likely to be successful also when delays are variable *or unknown to the attacker.* This means that we can use fixed delay analysis in order to choose the best attacker when delays are varied.

Another interesting observation from Figure 6 is that when delays are fixed, the damage is almost always lower than when delays are variable. This is because with variable propagation delays, the shortest paths are more diverse and include, on the average, more hops. Obviously, longer routing paths increase the probability of traversing a routing loop.



(a) flows caught by forwarding loops

(b) flows traversing loops of other flows

Fig. 6. The correlation between the damage when delays are equal and when they are variable

VII. Conclusions

We presented a new partition attack on link-state routing protocols in which a compromised router sends different LSAs to different neighbors. This enables a single compromised router to prevent the other routers from building a correct and consistent picture of the network topology. Consequently, forwarding loops are created, the length of some routing paths in the network is increased, and some pairs of routers become disconnected. Most importantly, this attack cannot be prevented even if LSAs are encrypted and digitally signed with current schemes.

We showed that this attack can divert much traffic to forwarding loops, and degrade most of the traffic in real network topologies. We also showed a simple method for evaluating which routers would be most efficient in generating an attack. Finally, we proposed some mitigation methods, though these require changes to specifications of existing link-state routing protocols.

References

[1] A. Barbir, S. Murphy, and Y. Yang. Generic threats to routing protocols. RFC 4593, October 2006.

[2] B. R. Bellur and R. G. Ogier. A reliable, efficient topology broadcast protocol for dynamic networks. In *INFOCOM*, 1999.

[3] M. Bhatia, V. Manral, M. Fanto, R. White, M. Barnes, T. Li, and R. Atkinson. OSPFv2 HMAC-SHA cryptographic authentication. *RFC 5709*, October 2009.

[4] H. Y. Chang, S. F. Wu, and Y. F. Jou. Real-time protocol analysis for detecting link-state routing protocol attacks. *ACM Transactions on Information and System Security (TISSEC)*, 4(1):1–36, February 2001.

[5] T. H. Clausen and P. Jacquet. Optimized link state routing protocol (OLSR). RFC 3626, October 2003.

[6] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta. VL2: a scalable and flexible data center network. In *ACM SIGCOMM*, pages 51–62, 2009.

[7] D. J. Haglin and S. M. Venkatesan. Approximation and intractability results for the maximum cut problem and its variants. *IEEE Transactions On Computers*, 40:110–113, January 1991.

[8] R. Hauser, T. Przygienda, and G. Tsudik. Lowering security overhead in link state routing. *Computer Networks*, 31(8):885 – 894, 1999.

[9] E. Jones and O. L. Moigne. OSPF Security Vulnerabilities Analysis. Internet-Draft draft-ietf-rpsec-ospf-vuln-02, IETF, June 2006.

[10] B. Kumar. Integration of security in network routing protocols. *SIGSAC Rev.*, 11(2):18–25, April 1993.

[11] J. Moy. OSPF Version 2. RFC 2328, April 1998.

[12] S. Murphy and M. Badger. Digital signature protection of the OSPF routing protocol. In *Proceedings of the Symposium on Network and Distributed System Security*, pages 93 – 102, 1996.

[13] S. Murphy, M. Badger, and B. Wellington. OSPF with digital signatures. RFC 2154, June 1997.

[14] G. Nakibly, D. Gonikman, and A. Kirshon. Owning the routing table - new ospf attacks. Technical report, Black Hat USA, 2011.

[15] G. Nakibly, D. Gonikman, A. Kirshon, and D. Boneh. Persistent OSPF attacks. In *Proceedings of NDSS*, 2012.

[16] G. Nakibly, E. Menahem, A. Waizel, and Y. Elovici. Owning the routing table - part ii. Technical report, Black Hat USA, 2013.

[17] G. Nakibly, A. Sosnovich, E. Menahem, A. Waizel, and Y. Elovici. OSPF vulnerability to persistent poisoning attacks: A systematic analysis. In *Proceedings of ACSAC*, 2014.

[18] P. Papadimitratos and Z. Haas. Secure link state routing for mobile ad hoc networks. In *Proceedings of Symposium on Applications and the Internet Workshops*, pages 379–383, January 2003.

[19] S. Rai, B. Mukherjee, and O. Deshpande. IP resilience within an autonomous system: current approaches, challenges, and future directions. *IEEE Communications Magazine*, 43(10):142 – 149, 2005.

[20] C. Santivanez and R. Ramanathan. Hazy sighted link state (HSLS) routing: a scalable link state algorithm. *BBN Technologies, Cambridge, MA, Tech. Rep. BBN-TM-1301*, 2001.

[21] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson. Measuring ISP topologies with rocketfuel. *IEEE/ACM Transactions on Networking*, 12(1):2–16, 2004.

[22] B. Vetter, F. Wang, and S. Wu. An experimental study of insider attacks for OSPF routing protocol. In *Proceedings of the International Conference on Network Protocols*, pages 293 –300, October 1997.

[23] F. Wang, F. Gong, F. Wu, and R. Narayan. Intrusion detection for link state routing protocol through integrated network management. In *Proceedings of 8th International Conference on Computer Communications and Networks*, pages 634–639. IEEE, 1999.

[24] F. Wang, B. Vetter, and S. F. Wu. Secure routing protocols: Theory and practice. Technical report, North Carolina State University, May 1997.

[25] F. Wang and S. Wu. On the vulnerabilities and protection of OSPF routing protocol. In *Proceedings of 7th International Conference on Computer Communications and Networks*, pages 148 –152, 1998.