TRACING BYZANTINE FAULTS IN AD HOC NETWORKS*

Mike Burmester, Tri Van Le and Matt Weir
Department of Computer Science,
Florida State University, Tallahassee, Florida 32306, USA

Abstract

Ad hoc networks are collections of mobile nodes with links that are made or broken in an arbitrary way. They have no fixed infrastructure and, usually, have constrained resources. The next generation of IT applications is expected to rely heavily on such networks. However, before they can be successfully deployed several major security threats must be addressed. These threats are due mainly to the ad hoc nature of such networks. As a result it may be much harder (or even impossible) to establish routing paths that can tolerate Byzantine faults. Recently a Byzantine faults tracing protocol has been proposed. This combines a reliability metric based on passed history with an adaptive probing technique. In this paper we first show that such an approach is fundamentally flawed and cannot be used to detect malicious faults. We then propose a new tracing algorithm that exploits a basic property of broadcast channels, ping channels, and authentication mechanisms, to locate Byzantine faults.

Keywords

Ad hoc networks, Byzantine faults, fault-tracing.

1. Introduction

Ad hoc networks are dynamic collections of self-organizing mobile nodes with links that are changing in an unpredictable way. They are characterized by a dynamic topology and the lack of any fixed infrastructure. The communication medium is broadcast. The nodes can be regarded as wireless mobile hosts with limited power, range and bandwidth. The recent rise in popularity of mobile wireless devices and technological developments has made possible the deployment of such networks for several applications. Indeed, because ad hoc networks do not have any fixed infrastructure such as stations or routers, they are highly applicable to emergency deployments, disasters, search and rescue missions and military operations. So far, most of the research has focused on functionality issues and efficiency (see e.g., [2,5,8,10,9,14,16,17,18,19,21], with security given a lower priority, and in many cases, regarded as an add-on afterthought technology rather than design feature (e.g. [12,15]).

Our goal in this paper is to show that tracing malicious (insider) faults of ad hoc networks is not as simple as it may appear at first. More specifically, that a Bayesian approach that exploits past statistical behavior (history) cannot be used to trace a malicious user (an insider) from his/her behavior, because such a user can avoid detection by behaving non-maliciously whenever an Intrusion Detection mechanism is triggered. In particular we show (Section 2) that the adaptive probing technique used in Auerbuch et al. [1] is flawed and that it will fail to trace a truly Byzantine malicious fault. We then propose our own tracing algorithm that combines cryptographic mechanisms with broadcast channels and ping channels.

2. The Ad hoc On-Demand Distance Vector (AODV) protocol

On-demand routing protocols are most appropriate for ad hoc networks due to their inherently dynamic nature. The Ad hoc On Demand Distance Vector protocol [15] is a commonly used routing protocol in which routes are only maintained as long as they remain active. This limits the overhead required to support route discovery, as every change to the network topology does not need to be broadcast to the entire network unless it affects an active route.

In the AODV protocol, the route setup is done in two phases. First, the network is flooded by the source node *s* with a request for a path to the destination node *d*. The body of the request body_s contains a source node identifier id_s, a destination node identifier id_d, a request sequence number and an authenticator $h_{Ksd}(\mathsf{body}_s)$ where h_{Ksd} is a keyed MAC (Message Authentication Code [18]) with key K_{sd} which is shared by *s* and *d*. Furthermore, the request also contains the hop count to be used by the destination node. In this way, the whole network can be mapped to a graph tree whose root is the source node *s* [7].

Since ad hoc networks are constantly changing, the source needs to closely monitor the path in order to maintain it in the face of transmission losses due to either network failures or malicious activities. When failure occurs, the source needs to identify the faulty links in

* This material is based on work supported in part by the U.S. Army Research Laboratory and the U.S. Research Office under grant number DAAD19-02-1-0235.

order to avoid using them in the construction of new paths. However, the problem is harder than it first appears because, not only can malicious nodes cause link failures under their control, but they can also damage other links not under their control by supplying false information to the source and destination nodes.

3. How not to solve the problem

A Bayesian approach can only be used to trace faults that follow a similar pattern to what happened in the past. Therefore, an intrusion detection mechanism that exploits past statistical behavior can only be used to trace faults that follow a predictable pattern.

By definition, Byzantine faults do not follow any pattern, and may be strongly dependent on other events under the control of the adversary. So although intrusion detection methodologies may succeed in tracing "copycat faults", they will not trace a determined adversary who will use an unpredictable strategy.

In Awerbach et al [1] faults are determined by the loss rate they cause; this must be above a certain threshold. This approach can be used to distinguish actual faults from faults caused by link breaks due to the ad hoc-ness of the network. The Tracing protocol in [1] uses a two-prong analysis on routing paths. First a routing path that exhibits an intolerable loss rate (above the threshold) is selected. Then the "faulty" path is adaptively probed to locate the faulty nodes by using a divide-and-conquer strategy.

The justification for this protocol seems to be that since faults follow a certain predictable pattern they can be distinguished from the faults that are caused only by the ad hoc-ness of the network for which, presumably, the loss rate is much lower. The assumption seems to be that the loss rate due to the ad hoc-ness is uniformly distributed and that the additional loss rate caused by malicious behavior can be used to tip the "threshold-loss" scale and trigger the tracing procedure.

There are several major faults with this reasoning. The main one is that, as pointed out earlier, Byzantine faults do not follow any particular pattern. Another is that that the assumption of a uniform ad hoc-ness is unjustified, and very restrictive to say the least. However let us assume for the moment that the adversary, in a moment of relapse, has used a copycat attack and that the path he controls is going to be probed by the tracing algorithm for malicious faults. Since the probing protocol in Awerbuch et al.[1] is different from the ordinary communication protocol, one must assume that adversary will become aware that he is now under investigation. A cunning adversary (Byzantine adversaries are cunning) would now behave faultlessly, making it impossible to be traced. Since the probing protocol is quite expensive (the probing protocol may require up to $\log n$ rounds of communications, where n is the length of the route [18]), one could regard this as a successful Denial of Service attack by the adversary.

4. A new Byzantine faults tracing protocol

Our tracing protocol will exploit the *broadcast channel assumption*, *ping* (*heart beat*) channels and an *authentication mechanism*:

- **The broadcast channel assumption:** All nodes within one broadcast-hop range to listen to each other. In particular, if nodes *A* and C are one hop neighbors of node *B*, then node *A* can listen to all communication from *B* to *C*.
- **The ping channel:** This is established by having each node broadcast at regular intervals a ping when there are network activities.
- **The authentication mechanism:** This makes it possible for all the nodes to authenticate their messages. Depending on the tracing requirements we will either use a keyed MAC or a digital signature (such as the RSA cryptosystem [18]).

It is easy to see that by combining these primitives, it is possible to trace all ad hoc link faults which involve at least one non-faulty node, provided that there are fault-free paths linking any two nodes. Indeed, if a link is broken then the adjacent nodes will become aware of this since they will not receive each other's pings. They can then send an authenticated broken link report to the sender and receiver on any route, which uses this link. Alternatively, the broken link may be fixed as in AODV [15] and a report sent to the sender node providing the updated path length. A broken link between malicious nodes is of little relevant since it is, in any case, under control of the adversary.

The tracing algorithm

We shall assume that a routing path has already established by using the Ad hoc On-demand Distance Vector (AODV) routing algorithm (see e.g., [15]), but will modify the transmission protocol to allow for traceability as follows.

Let $R = (s=x_0,x_1,\ldots,x_r,d)$ be the routing path, where s and d are the source and destination nodes. The sending phase is essentially the same as that of the AODV protocol, except that we require that the source s and each intermediate node x_i set a timer_i to a time based on their distance from the destination.

Additional requirements for the AODV protocol

On the upstream phase

$body_s := (id_s, id_d, counter_d, data), h_{Ksd}(body_s)$;

$timer_i := 2(t-i)\tau$;

Where τ is an upper bound on the time allocated for each hop.

On the downstream phase:

$body_d := (id_s, id_d, counter_d), h_{Ksd}(body_d)$;

If a node x_i does not receive an acknowledgement, $body_d$, or a valid fault_report in the allocated time, $timer_i$, then the node x_i reports the fault to the source node *s* by sending an authenticated fault_report:

$fault\_report_i := (id_s, id_d, counter_d, id_{xi}), sig_{Ki}(fault\_report_i)$

Tracing faults

If the source *s* receives a valid $fault\_report_j$, then either x_i or x_{i+1} is a faulty node. If *s* receives no valid report within the allocated time τ, then x_1 is the faulty node. Observe that τ must be significantly less than the expected lifetime of each hop. Furthermore, one must allow for some time for nodes to process/forward packets, so that the link between nodes x_i and x_{i+1} does not break (because of the ad hoc nature of the network) before x_{i+1} forwards its packet.

This Tracing algorithm will succeed in tracing a Byzantine fault because of our assumption that all the links on the route are not faulty, and because if both x_i and x_{i+1} were both non faulty then x_i would not authenticate a fault report. The fault will be attributed to *both* x_i *and* x_{i+1}.

Remarks

- Our Tracing algorithm will trace *pairs* of linked nodes, so that with each faulty node an innocent node may be implicated. Note that it is impossible to decide which of the two nodes x_i and x_{i+1} is actually malicious from a single Byzantine attack. This is because there is no way of distinguishing which one of the two is the liar.
- Our Tracing algorithm is more efficient than the Awerbuch et al. tracing algorithm [1], for several reasons. One reason is that there is only one mode of operation and that tracing is only triggered if an actual fault occurs. So the adversary cannot employ a hide-and-seek strategy in which he/she is malicious only during the communication phase and not during the tracing phase. Another reason is that for each fault only one fault_report is needed, whereas in the Awerbuch et al. tracing algorithm [1] up to $\log n$ fault_reports may be needed (depending on the strategy of the adversary), where *n* is the length of the routing path.
- Our Tracing algorithm which is described for routes determined by an AODV routing algorithm can easily be extended to routes determines by any other routing algorithm.

5. Conclusion

We have presented a Byzantine faults tracing algorithm that can be used to trace malicious behavior. This algorithm does not rely on any stochastic analysis, and is not restricted to "copycat attacks" or attacks that follow a certain pattern. The faults are attributed to pairs of adjacent nodes, and therefore may implicate non-faulty nodes. However this may not be an excessive price to pay, if one takes into account the damage to the whole network a Byzantine fault may cause. Alternatively, one could use traced faults from several attack instances to pinpoint the actual malicious nodes. This would force the adversary to significantly restrict the number of Byzantine attacks, unless of course the adversary wants to sacrifice some nodes under his/her control.

References

[1] B. Awerbuch, D. Holmer, C. Nita-Rotaru and H. Rubens. "An On-Demand Secure Routing Protocol Resilient to Byzantine Failures", *ACM Workshop on Wireless Security (WiSe'02),* September 2002.

[2] E.M. Belding-Royer and C.-K. Toh. "A review of current routing protocols for ad-hoc mobile wireless networks". *IEEE Personal Communications Magazine*, pages 46–55, April 1999.

[3] J. Broch *et al*. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *Proc. ACM MOBICOM*, pp. 85–97, 1998.

[4] C.C. Chiang et al. "Routing in Clustered Multihop, Mobile Wireless Networks with Fading Channel", *Proceedings of IEEE SICON* '97, pp. 197--211, April 1997.

[5] C. R. Davis. IPSec: *Securing VPNs.* McGraw-Hill, New York, 2000.

[6] R. Dube, C.D. Rais, K.Y. Wang, and S.K. Tripathi, "Signal Stability based Adaptive Routing for Ad-Hoc Mobile Networks", *IEEE Personal Communications*, pp. 26--45, 1997.

[7] L.R. Ford and D.R. Fulkerson. *Flows in Networks*. Princeton University Press, Princeton, NJ, 1962.

[8] J.-P. HuBaux, L. Buttyan, and S. Capkun. The quest for security in mobile ad hoc networks. In *Proc. ACM MOBICOM*, 2001.

[9] D.B. Johnson and D.A. Maltz, "Dynamic Source Routing in Ad-Hoc Wireless Networks", *Mobile Computing*, ed. T. Imielinski and H. Korth, Kluwer Academic Publisher, pp. 152--181, 1996.

[10] Y.B. Ko and N.H. Vaidya, "Location-Aided Routing in Mobile Ad Hoc Networks", *Proceedings of ACM/IEEE MOBICOM* '98, October 1998.

[11] J. Kong et al., Providing robust and ubiquitous security support for mobile ad-hoc networks. In *Proc. IEEE ICNP*, pp. 251–260, 2001.

[12] S. Murthy and J.J. Garcia-Lunca-Aceves. An efficient routing protocol for wireless networks. *ACM Mobile*

Networks and Applications Journal, pp. 182–197, Oct. 1996.

[13] V. Park and M. Corson. A highly adaptive distributed routing algorithm for mobile wireless networks. In *Proc. INFOCOMM*, April 1997.

[14] C.E. Perkins and P.Bhagwat, "Highly Dynamic Destination-Sequenced Distance-Vector Routing for Mobile Computers", *Computer Communications Review*, pp. 224--244, October 1994.

[15] C.E. Perkins and E.M. Royer. Ad hoc on-demand distance vector routing. In *IEEE Workshop on Mobile Computing Systems and Applications*, pp. 90–100, Feb. 1999.

[16] K. Sanzgiri *et al*, "A Secure Routing Protocol for Ad Hoc Networks". citeseer.nj.nec.com/sanzgiri02secure.html

[17] S. Signh, M. Woo, and C.S. Raghavendra, "Power-Aware Routing in Mobile Ad Hoc Networks", Proceedings of ACM/IEEE MOBICOM '98, October 1998.

[18] B. Schneier. *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. John Wiley & Sons, Inc, New York, 1996.

[19] C.K. Toh, "A Novel Distributed Routing Protocol To Support Ad-Hoc Mobile Computing", Proceedings of 1996 IEEE Fifteenth Annual International Phoenix Conference on Computers and Communications, pp. 480--486, March 1996.

[20] S. Yi, P. Naldurg, and R. Kravets. Security-aware ad hoc routing for wireless networks. In *Proc. ACM Mobihoc*, 2001.

[21] L. Zhou and Z. J. Haas. Securing ad hoc networks. *IEEE Network*, 12(6):24–20, 1999.