Secure routing built on BATMAN-ADV

2016-12-10

1 Work in progress

This paper is incomplete, feel free to provide feedback via Github issues

Contents

2 Abstract

In mesh networks packets are routed by participants in the network in a distributed manner. To make these networks feasible the amount of information spread to each node about the total state of the network must be kept small. In overlay networks, networks where any node may communicate with any node freely without concern for layer 1 and 2 connectivity, security against dishonest or malicious nodes is a common practice.

For mesh networks that are not overlay networks, operating at a local level rather than over the wider internet the ability to handle malicious nodes is fairly rare and none exist that can provide the same properties as widely used distributed software for overlay networks [9]. Meaning the ability to operate without a central authority of any kind in a manner significantly resistant to hostile actors.

In this paper we present a concept to secure the distance vector metric in BATMAN-ADV against hostile actors, allowing the network to continue to function in the face of attacks so long as a route of participant nodes to a given destination exists.

3 Batman-Adv

Batman-Adv is the layer 2 version of the original BATMAN protocol outlined in [1]. Conceptually the protocol is very simple. Every originator period every node participating in the network broadcasts an originator message containing a MAC address, which identifies the node that 'originated' the message as well as the MAC address of the last node to forward it. When an originator message is broadcast every node that receives the broadcast updates the expanded bandwidth field with it's own estimation of the bandwidth between itself and its neighbor if and only if the predicted bandwidth is lower than the value already in the originator [3].

As the originator messages spread through the network each node checks each originator message it receives against it's own local routing table, either to be added or updated if the throughput advertised by the received originator message is better or the sequence number of the message is higher. Since originator messages contain the last hop that forwarded the message this data combined with throughput and sequence number is used to maintain an up to date routing table with the best first hop along the path to any other node in the network.

To send data to any other node it is sent to the neighbour node listed in the local routing table, who will forward that data in turn based on it's own copy of the routing table. This process continues until the data is delivered to its destination.

Batman-Adv is implemented as an up-stream Linux kernel module with several optimizations from the original concept, including network coding to batch the broadcast of originator messages and several optimizations focusing on allowing hybrid networks containing traditional Ethernet links as well as wireless links [2]. Perhaps the most promising feature of Batman-Adv is that

is remains competitively performant when compared to other mesh protocols [4]

4 Potential Attacks

- Black Hole attack, a route pretends to have a routing metric of infinity or zero, whichever is better, attracting all network traffic through itself, which it then never forwards, causing all communication to be lost.
- Spy attack, the same as the Black Hole except packets are honestly forwarded. Allowing a single node to spy on all other nodes.
- Impostor attack, a malicious node pretends to be a highly sought after destination or route participant, thus receiving traffic they should not have otherwise received.
- Slight of hand attack, a node changes it's own routing metric only slightly, gathering significantly more traffic than it should but not all of the traffic.

5 Modifications to Batman-Adv protocol spec

5.1 Message signing

To prevent impostor attacks all packets will be signed with a symmetric encryption key distributed in the originator message of a node. In well connected networks man in the middle attacks will be impossible to execute without the victim being alerted by honest nodes. For a poorly connected node man in the middle attacks are preventable by storing the identities of previously seen nodes or communicating with a trusted mobile node.

5.2 Circular Ack

To prevent the remaining set of attacks we must provide some independent way for participants ina connection to verify the authenticity of the route throughput metric. This method must take into account that asymmetric routes are possible, so in a connection from node A to node B it's possible that the nodes sending traffic to B will never see any traffic B sends to A.

We propose a concept we call a ‘circular ack‘ because of the way it will circle and entire asymmetric routing loop and provide route quality infor-

mation from both sides to both sides. At a regular period each endpoint of a route will create and send a message containing the following information.

- A timestamp
- A ack period number
- How many valid packets have been received from the other endpoint during the ack period
- Number of packets sent to the other end point during the ack period
- Connection Saturated Flag
- signature for the above data
- The above data from the other participant from the previous ack period.

When each side of the connection receives the ack message, they compare the information about how much data was received with their records of how much data was sent. From this, they can estimate the throughput of their route to the other node and compare it to the advertised throughput on the connection.

As the data makes its 'circle' non-endpoint connection participants can view the data and compare it to their own records allowing them to form more accurate estimates of the throughput of both their outgoing and incoming links with at most a two ack period delay.

Finally the packet waiting counter can be used by all parties to determine if a connection is being fully utilized. If the connection is being fully utilized it is valid to update the throughput of the link to be the throughput being seen.

5.3 Accurately updating the saturation flag

TODO lots of research on this, how do packet buffers work between layer 2 and 3? The saturation flag can be computed as the minimum number of packets waiting across several layer 3 connections running over a single BATMAN-ADV layer 2 route. The goal is to ensure that the flag is only set when a connection within the mesh is being saturated and not when a connection outside of the mesh is being saturated. If a node is communicating with another node and the throughput rate does not match the data rate the packet buffers of each hop will be filled until all nodes in the route have

packets waiting on that specific link. In the case that a node is attempting to communicate with a slow server on the internet for example all packets will reach the exit node and wait there with other connections over that link keeping little to no backup.

5.4 Applying corrected metrics

Since we can not confirm if a node is lying or simply has a flawed method to estimate it's own link bandwidth the ack metric should be used to adjust the advertised throughput of a given route not for blacklisting or attempting to pushing malicious nodes. Any node that sees an applicable ack will adjust the advertised throughput for that route in it's own originator messages if appropriate. Meaning large swaths of the network will begin to participate in throughput correction in a single ack period and most nodes will be able to participate in several simultaneous throughput corrections without additional overhead beyond the packet counters required for the first correction.

Nodes not participating in a correction simply update the metrics they receive with their own throughput and rebroadcast the network as a whole will converge to the true throughput of a given connection, hopefully within no more than a few ack periods. The largest remaining question is for how long to continue to apply the adjusted metric after acks are no longer received. In the case of a malicious node advertising routes it will not complete, throughput will quickly correct to zero and that route will stop being advertised, only for the ack's to expire and the process to repeat.

In order to mitigate this an exponentially increasing time-out will be used for bandwidth corrections beyond a small tolerance. A node that has participated in a correction will record the time it was last performing a correction for a given route T_{last} and the size of the correction δ. When participating in another correction on the same route the amount of time to apply the bandwidth correction after the ack has expired will be determined as.
$T_{adjustment} = \frac{C_1}{T_{now} - T_{last}}^{C_2 \delta}$

Where C_1 and C_2 are constants to be adjusted and hardcoded. The goal of this formula is to provide a longer adjustment time based on nodes stopping quickly returning to adjusting the same route and further increased by the magnitude of the route adjustment since these characteristics fit the expected profile of a malicious node that can't provide any actual use to the network.

5.5 Malicious acks

Now that we have a system to prevent malicious advertisement of false throughput how do we prevent malicious advertisement of false throughput using the correction mechanism. This actually isn't a real problem, the key to understanding why is realizing that each originator represents the best route to a given node and has no bearing on the best route to any other given node even though they will often share physical links. Lying about your throughput to the other endpoint is easily falsifiable, intermediate nodes know how many packets they actually received and forwarded and the ack is signed so the value the other endpoint returns can't be modified. The only thing a node can lie about in a ack message is to cause it's own throughput to be adjusted downwards.

References

[1] David Johnson, Ntsibane Ntlatlapa, and Corinna Aichele. *A simple pragmatic approach to mesh routing using BATMAN* (2008)

[2] Cigno, R., and Daniele Furlan. *Improving BATMAN Routing Stability and Performance* (2011)

[3] Quartulli, Antonio, and Renato Lo Cigno. *Client announcement and Fast roaming in a Layer-2 mesh network* (2011)

[4] Murray, David, Michael Dixon, and Terry Koziniec. *An experimental comparison of routing protocols in multi hop ad hoc networks* Telecommunication Networks and Applications Conference (ATNAC), 2010 Australasian. IEEE, 2010.

[5] Britton, Matthew, and Andrew Coyle. *Performance analysis of the BATMAN wireless ad-hoc network routing protocol with mobility and directional antennas* Military Communications and Information Systems Conference (MilCIS), 2011. IEEE, 2011.

[6] Hu, Yih-Chun, David B. Johnson, and Adrian Perrig. *SEAD: Secure efficient distance vector routing for mobile wireless ad hoc networks* Ad hoc networks 1.1 (2003): 175-192.

[7] Perrig, Adrian, et al. *SPINS: Security protocols for sensor networks* Wireless networks 8.5 (2002): 521-534.

[8] Hu, Yih-Chun, Markus Jakobsson, and Adrian Perrig. *Efficient constructions for one-way hash chains* International Conference on Applied Cryptography and Network Security. Springer Berlin Heidelberg, 2005.

[9] Lundberg, Janne. *Routing security in ad hoc networks* Helsinki University of Technology, http://citeseer. nj. nec. com/400961. html (2000).