Routing Attacks as a Viable Threat: Can Software Systems Protect Themselves?

Dan Alistarh[1] Gilles Trédan[2] Ioannis Avramopoulos[2] Petr Kuznetsov[2]

[1] Swiss Federal Institute of Technology, Lausanne, Switzerland
[2] Deutsche Telekom Laboratories/TU Berlin, Berlin, Germany

Abstract

In this paper, we show that distributed systems are vulnerable to routing attacks and propose an architecture to obviate this vulnerability. A somewhat surprising finding is that even a small-scale routing attack can completely disrupt the operation of a state-machine replication service. The architecture that we propose is based on the following simple ideas: (1) Circumvent the adversary if possible and (2) if it is not possible, relax the application semantics.

1 Introduction

Recent incidents (e.g., [2]) indicate that *network-layer attacks* exploiting the routing protocol can be detrimental for the availability of *application-layer* distributed software systems. Such network-layer attacks can be performed from peripheral autonomous systems through, for example, hijacking the address space of the software service.[1] Therefore, it is natural to ask to what extent distributed systems are vulnerable to these attacks and to what extent they can be protected. Although it is true that a significant part of research in distributed systems has been dedicated to making these systems robust to failures, we obtain a perhaps surprising result that the state-of-the-art techniques to ensure robustness do not suffice to ensure the system's availability.

A standard technique to achieve fault-tolerance in distributed systems is to replicate data and computation on multiple locations so that, even if one or more replicas fail, the system will be able to provide correct and continuous service as long as enough replicas remain operational. In this context, a lot of attention has been given to providing robustness in the so-called *Byzantine* fault model [9], according to which faulty replicas behave in an arbitrary fashion. It would seem that a Byzantine fault-tolerant (BFT) system should also be robust against routing attacks. However, although a routing attack cannot violate the system's correctness, we show that it can easily prevent the system from making progress. The reason can be traced to the fact that existing BFT replication protocols (e.g., [5]) assume that communication links are *reliable* (implying, for instance, that every pair of correct replicas are able to communicate).

Having identified this threat, we present a system, which we call RBFT (Robust BFT), that is designed to operate against it. The system architecture is based on two simple ideas: (1) Circumvent the adversary if possible, and (2) if it is not possible, relax the application semantics. In this paper, we investigate the robustness of this architecture using simulation (omitting the analytical results in the interest of space). One of our findings is that network support beyond the scope of a software system's application-specific resources can be instrumental in improving robustness.

Related Work. Much previous work in defending against routing attacks (e.g., [7]) has focused on network-wide network-layer countermeasures. In this paper, we combine more easily deployable network-layer countermeasures with application-layer ones.

Our countermeasures include overlay routing and anycast. Using overlays to overcome routing faults has been explored in systems such as RON [3], Detour [10], and ACR [12]. The ability of anycast to protect user access to backbone DNS servers from routing attacks is evaluated in [4]. In contrast, we are not only interested in protecting the ability of users to access a general replicated system, but also in the availability of the replica consistency protocol. Previous work in detecting routing attacks (such as iSPY [13]) is in principle complementary to our approach.

Finally, the idea of relaxing the application semantics in defending against routing attacks has not been explored before to the extent of our knowledge.

2 BFT is vulnerable to routing attacks

We consider a snapshot of the Internet's autonomous-system graph annotated with routing policies, taken from CAIDA (www.caida.org), on which we simulate routing attacks using the BSIM BGP simulator [1]. We select the p participating nodes and the b attacking nodes at random from the set of autonomous systems. In this setting, a necessary (but not sufficient) condition for BFT to make progress is the existence of a *star*, that is, a subgraph of the overlay

[1] This is what happened when Pakistan Telecom claimed to hold YouTube servers (http://news.cnet.com/8301-10784_3-9878655-7.html), which resulted in a more than two hours blackout of YouTube's operation.

graph consisting of one overlay node connected to $2f$ other nodes by bidirectionnal links that have not been hijacked, where $f < p/3$ is the maximal number of faulty replicas and p is the number of replicas.

Figure 1(a) measures the impact of an attack on the replica overlay. Each point of the curves is the average of 160 independent experiments. Our security performance metric is the *liveness ratio*, defined as the percentage of experiments in which the overlay contains a star. The empirical liveness ratio is shown as a function of b, for 10, 20 and 40 servers. For instance, the liveness ratio is lower than 20% assuming $b \geq 4$, independently of p. This clearly highlights the significance of the threat as a system that consists of $40 = 3f + 1$ replicas is expected to tolerate $13 = f$ Byzantine replicas in the absence of routing attacks. Finally, note that the availability benefit gained by increasing the number of replicas is negligible.

3 RBFT Architecture

To overcome routing attacks, we try to ensure that clients and servers have enough redundant paths to circumvent the adversary through protocols designed for this purpose. If this is not possible, the system gracefully degrades by changing the application semantics.

3.1 Obtaining Path Diversity

In the Internet, every router selects at most one path for each destination, and routing policies restrict what paths can be selected. Moreover, applications do not typically participate in the routing decisions. Therefore, the available default paths are rather restricted, and one of our challenges is to improve path diversity. To that end, we rely on the following methods. (1) We try to *squeeze* as many paths as possible from a given system configuration (where by *configuration* we mean a given placement of replicas to autonomous systems). (2) We place the replicas *strategically*. (3) We *reconfigure* the system dynamically by adapting the number of replicas and their network location.

To squeeze paths from a given configuration, we rely on two tricks. The first trick is to use *overlay* paths: if two replicas cannot communicate directly, they try to communicate through a one or more intermediate replicas. The second trick is to expose the network provider's path diversity to the application. That is, if a provider has more than one path to a particular (destination) replica, a (source) replica being hosted at this provider's network would be able to use any of those paths.

The performance of a particular configuration against a routing attack depends on both the number of autonomous systems where replicas reside as well as the location and role of those autonomous systems. For example, *tier-1* autonomous systems, which do not have providers, are able to contribute significantly higher path diversity than *stub* (peripheral) autonomous systems, and it would be beneficial for the robustness of the service to host its replicas inside such a network. This intuition is verified in our simulations.

Finally, if the available path diversity is insufficient to ensure progress, we change the system configuration by adding, removing, and relocating replicas (leveraging the cloud resources available in the Internet). Finding a new configuration and making sure that the configuration is properly adopted by the system in the presence of a routing attack can partly be a very interesting application for ideas summarized in [8].

3.2 Exploiting the Path Diversity

Once redundant paths are available, carefully designed protocols should be in place to circumvent the adversary. We care about the communication between the replicas and the clients as well as the communication among the replicas.

Client-to-Replica Paths. There are two aspects related to protecting communication between the replicas and the clients: making sure that the client requests reach the replicas and making sure that the replica responses reach the clients. We only discuss the second aspect (the techniques for client-to-replica communication are similar).

In BFT, upon execution of a request, each replica forwards the corresponding response to the corresponding client *directly*, and the client accepts the response if it receives identical responses from $f + 1$ replicas (one of them should come from a correct replica). However, because of a routing attack, the response may not reach the client. As a result, the client may not receive $f + 1$ matching responses to return the response to the application. To overcome this issue, we require from each replica to broadcast its result to the rest of the replicas, and as soon as each replica collects $f + 1$ distinct matching responses, it sends them back to the client. As long as enough replicas are pairwise connected (an assumption that we will revisit in the next section) and a non-faulty path to the client exists, the client receives $f + 1$ correct responses.

Replica-to-Replica Paths. In BFT, broadcast is the basic primitive of communication. Since a routing attack can severely affect a significant number of direct paths in a naive broadcast protocol, one method to exploit fault-free overlay paths is to rely on *flooding*: if a replica receives a message m it has not seen before, it forwards m to every other replica. The resulting protocol ensures progress that as long as a quorum of correct replicas maintain a (eventually synchronized) connected overlay.

To overcome the $O(p^2)$ message complexity of this approach, we require that flooding happens on a *subgraph* of the replica overlay (that we will call the *dissemination graph*). In the best case, this method employs $O(p)$ transmissions per broadcast message (when the dissemination graph is a tree), however, we may not be able to ensure delivery to all replicas that are connected by a fault-free path to the source. In the following, we sketch a protocol that approximates the $O(p)$ complexity by adapting to the attack and selecting the "best" dissemination graph.

At a high level, the protocol works by having replicas *adaptively* decide on the dissemination graph using performance feedback. The adaptation procedure is based on two components: A performance monitor and a graph selection algorithm. The performance monitor determines for each request whether it was serviced successfully or not. The

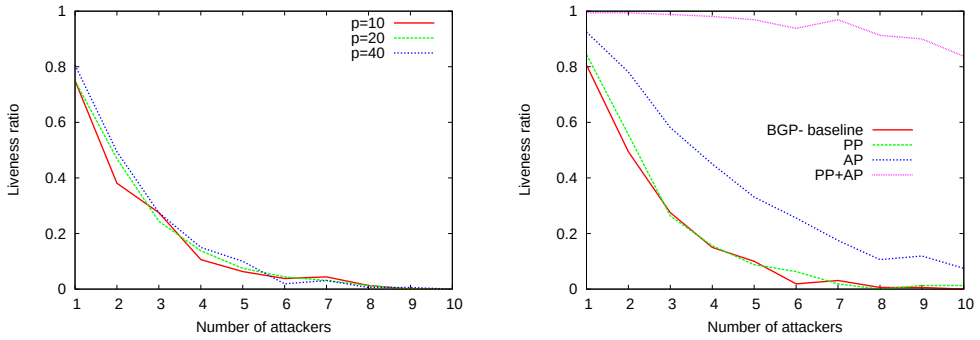

Figure 1. (a) Impact of a routing attack on BFT. (b) Effect of various defense strategies on the resilience of RBFT.

output of the performance monitor is fed to the dissemination graph selection algorithm, which is based on an online learning algorithm, e.g., Hedge [6], that assigns to each candidate dissemination graph a probability and performs the selection based on the resulting probability distribution.

3.3 Changing the Application Semantics

Another way of adapting to routing attacks is to trade off the strong consistency properties of BFT for availability, extending the approach initiated by Zeno [11]. Roughly, each group of $f + 1$ correct replicas in the overlay resulting after a routing attack would provide a *weakly* consistent BFT service. Once a partition is repaired, the divergent views of the distributed system state maintained by the components are automatically merged in a common consistent view. Note that, since Zeno only generalizes BFT's semantics, it can work in concert with the other parts of the system.

4 Evaluation

In this section, we evaluate the performance of RBFT through simulations. We first measure to what extent the replica overlay can resist the routing attack, assuming that replica-to-replica communication proceeds over overlay paths. We use the setup of Section 2, i.e. a snapshot of the real Internet topology and 40 replicas. We analyze the security performance of RBFT when only overlay paths are used (the baseline), and when two additional strategies for enhancing RBFT are employed. The first additional strategy, which we call PP (Priority Placement), places some of the replicas at tier-1 AS nodes, i.e. at nodes that have no provider. The second additional strategy, called AP (Alternate Paths), further increases the path diversity, by allowing replicas to use *all* alternate BGP paths visible to their providers. Multipath is used here in an innocuous fashion as these alternate paths are not readvertised into BGP.

Figure 1(b) shows the results of the simulation assuming the replicas and the attackers are placed at random. The relatively poor security performance of RBFT when only overlay paths are used (BGP-baseline) is perhaps surprising. Also, adding tier-1 peers (the PP strategy) does not improve the liveness ratio. The alternate-paths (AP) strategy provides significant improvement. Moreover, when both strategies are combined (PP+AP), the system exhibits surprisingly high resilience, which suggests an additive power of these strategies. One intuition for this is that tier-1 AS'es are close to most servers, but also to many attackers. Allowing them to change routes reveals all the potential of their strategic placement.

References

[1] The BSIM BGP simulator. http://www.cs.unm.edu/ karlinjf.

[2] Six worst internet routing attacks. http://www.networkworld.com/news/2009/011509-bgp-attacks.html, January 2009.

[3] D. Andersen, H. Balakrishnan, F. Kaashoek, and R. Morris. Resilient overlay networks. *SIGOPS Oper. Syst. Rev.*, 35(5):131–145, 2001.

[4] I. Avramopoulos and M. Suchara. Protecting DNS from routing attacks: Two alternative anycast implementations. *IEEE Security and Privacy*, 7(5):14–20, Sept./Oct. 2009.

[5] M. Castro and B. Liskov. Practical byzantine fault tolerance. In *OSDI '99*, pages 173–186, Berkeley, CA, USA, 1999.

[6] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *JCSS*, 55(1):119–139, 1997.

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

[8] L. Lamport, D. Malkhi, and L. Zhou. Reconfiguring a state machine. *SIGACT News*, 41(1):63–81, 2010.

[9] L. Lamport, R. Shostak, and M. Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, 1982.

[10] S. Savage, T. Anderson, A. Aggarwal, D. Becker, N. Cardwell, A. Collins, E. Hoffman, J. Snell, A. Vahdat, G. Voelker, and J. Zahorjan. Detour: Informed internet routing and transport. *IEEE Micro*, 19(1):50–59, 1999.

[11] A. Singh, P. Fonseca, P. Kuznetsov, R. Rodrigues, and P. Maniatis. Zeno: Eventually consistent byzantine fault tolerance. In *NSDI '09*.

[12] D. Wendlandt, I. Avramopoulos, D. Andersen, and J. Rexford. Dont secure routing protocols, secure data delivery. In *In Proc. 5th ACM Workshop on Hot Topics in Networks (Hotnets-V)*, 2006.

[13] Z. Zhang, Y. Zhang, Y. C. Hu, Z. M. Mao, and R. Bush. iSPY: Detecting IP prefix hijacking on my own. In *SIGCOMM '08*, 2008.