PRESERVING PERFORMANCE OF BYZANTINE FAULT TOLERANT REPLICA GROUPS IN THE PRESENCE OF MALICIOUS CLIENTS

Sajeeva L. Pallemulle, Haraldur D. Thorvaldsson, Kenneth J. Goldman
Department of Computer Science and Engineering
Washington University in St. Louis
email: {sajeeva, harri, kjg}@cse.wustl.edu

ABSTRACT

The Castro and Liskov Byzantine Fault Tolerance protocol for replicated state machines (CLBFT) provides a practical means of tolerating arbitrary replica failures in replicated passive data servers. For better performance, CLBFT uses Message Authentication Codes (MAC) instead of public key cryptography to authenticate messages and preserves replica consistency even in the presence of malicious clients. However, CLBFT is susceptible to potential attacks by malicious clients using corrupted MACs to force replica groups into expensive configuration changes repeatedly. While not affecting correctness, this vulnerability can seriously impair the performance of the replica group. We propose modifications to CLBFT that address this problem. We identify two key forms of attacks and present a viable solution to each.

Keywords: Byzantine Fault Tolerance, Message Authentication Code, Malicious Client

1 Introduction

Critical distributed systems must employ Byzantine fault tolerance to make progress in the presence of arbitrary failures. The Castro and Liskov Byzantine Fault Tolerance protocol (CLBFT) [1] is a practical means of tolerating such failures. Unlike other approaches to this problem [2, 3] that use public key cryptography for message authentication, CLBFT uses message authentication codes (MAC) [4] to authenticate communication between hosts. The use of MACs drives down the cost of authentication substantially, thereby increasing the suitability of CLBFT for practical systems.

However, MACs can only be used to authenticate point to point communication, unlike public key cryptography that supports third party message verification. Inspite of this limitation, the CLBFT algorithm is resilient to malicious failures in both replicas and clients, in terms of correctness. However, a faulty client may utilize this weakness to force expensive configuration changes in replica groups repeatedly. The time to complete each subsequent configuration change increases exponentially. Therefore this vulnerability can seriously impair the performance of the algorithm. In this paper we identify two forms of attacks that can be initiated by malicious clients on replica groups and propose modifications to CLBFT to eliminate each threat.

The rest of this paper is structured as follows. Section 2 introduces the CLBFT algorithm and particularly the *view change protocol* that is used for configuration changes. In Section 3 we introduce two forms of attacks on replica groups by malicious clients. We present our modifications to CLBFT to avoid these attacks in Section 4. Section 5 argues the merits of our approach against other possible solutions. In Section 6 we conclude with a discussion of future work.

2 Background

The CLBFT algorithm uses $3f + 1$ replicas, where at most f can be faulty. Messages can be delayed, provided that the length of message delays does not increase faster than time (a weak assumption). MACs are used to verify authenticity of messages, and message digests [5] are used to reduce message size. The CLBFT algorithm for a mutating operation works roughly as follows (read operations require less communication). A client sends its request to a designated *primary* replica, which appends a sequence number and forwards it to the replicas in a *pre-prepare* message. The replicas use this message to enter the *pre-prepared* stage. Since the primary may be faulty, the replicas multicast a corresponding *prepare* message to each other, to ensure that all were given the same request and sequence number. Upon receiving $2f$ prepare messages matching the pre-prepare it received from the primary, a replica enters the *prepared* stage and multicasts a *commit* message to all the replicas. When it has matching commit messages from $2f+1$ replicas (possibly including itself), a replica enters the *committed* stage, executes the requested operation, and sends the result to the client. Upon receiving $f + 1$ matching replies, the client accepts that return value. If a client times out waiting for a reply (perhaps due to a faulty primary), it multicasts its original request to all the replicas. The replica starts a request timer if the operation has not yet executed. Also, if the replica has not yet received a preprepare, it forwards the request to the primary. When the operation completes, it replies to the client with the return value.

If progress under the current primary is unsatisfactory, the replicas change the primary in a *view change* operation. To propose a view change operation, the replicas first send out a *view-change* message including the digests of all requets that have *pre-prepared* and *prepared* but have not *committed*. It then starts a view timer to wait for the

completion of the view change. When a replica receives a valid view-change message it sends a *view-change-ack* message to the proposed primary to acknowledge receipt of the view-change message. When the new primary has at least *2f+1* view-change messages and at least *2f-1* (*2f+1* if its own message and the view-change message are counted) corresponding view-change-ack messages for each view-change message, it uses the information contained in the view-change messages to construct a *new-view* message. The new primary sends the new-view message to all replicas who in turn use that information to (possibly) update their local state and continue operation execution in the new view. If the view timer at a replica expires before a new view message is received, then the replica sends out a view change message requesting the next view in the sequence. To adapt view timers to prevalent network conditions the timeout values for both request timers and view timers double each time a replica decides to request a new view. Since CLBFT assumes that length of message delays does not increase faster than time, eventually all replicas are guaranteed to receive a new-view message.

3 Problem Description

In CLBFT, when a client sends a request to a replica group, it includes an *authenticator* which contains a MAC for each server replica. Each MAC is calculated over the digest of the message using a previously established shared secret key between the client and the corresponding replica. Regardless of whether the request was sent directly to the primary or multicast to the group, each server replica can use its MAC in the authenticator to verify the contents of the request, provided that its MAC has not been corrupted. However, faulty clients can include corrupt MACs in the authenticator and thereby prevent a subset of server replicas from verifying the request. Similarly a malicious server replica can also corrupt a subset of MACs to achive the same result. In CLBFT these cases are indistinguishable and this limitation leads to two scenarios where a malicious client can repeatedly force view change operations at the sever. [6]

3.1 Corrupted Non-Primary MACs

In the first form of attack, the malicious client includes a valid MAC for the primary but will include less than *f* other valid MACs for the rest of the non-faulty, non-primary replicas. Then the client sends the request to the primary who cannot detect that the MACs for the other replicas are invalid. The unsuspecting primary verifies the request as usual, assigns it a sequence number, and sends out the request and the corresponding pre-prepare message. Less than *f* other correct replicas are able to verify the request and therefore, send out prepare messages. The CLBFT algorithm specifies that a request can be verified either using the MAC or upon the receipt of *f+1* prepare messages. Neither case is possible in this attack and hence some non-faulty replicas will never send prepare messages for the request. Consequently, the sequence number assigned to the request will never be committed and hence no futher requests can be executed on the replica group. This situation will eventually lead to a view change operation. Furthermore, this attack can be repeated for the next primary by the malicious client.

3.2 Corrupted Primary MAC

In the second form of attack, the malicious client includes valid MACs for at least one non-faulty, non-primary replica and a corrupted MAC for the primary in its authenticator. It then directly sends the request to the corresponding non-primary replicas. Those replicas are able to verify the request and therefore they forward the request to the primary and start their request timers. The primary is unable to verify the request and hence cannot send out a pre-prepare message. The replicas that forwarded the request will eventually timeout and start a view change operation. This attack can also be repeated for the next primary by the malicious client.

4 Modifications

In this section we present modifications to CLBFT that will prevent each attack. In each case, the goal is to discriminate between a malicious client and a malicious primary.

4.1 Corrupted Non-Primary MACs

In this case, the view change is triggered by the inability to execute the request associated with a particular sequence number. Therefore, if the primary can learn whether at least *2f* other replicas can verify the request *before* assigning a sequence number to a request, this attack can be avoided. The simplest approach to this problem is for the primary to first multicast the digest of the request along with the authenticator to all replicas who will in turn send a *valid-request* message back to the primary. The primary will collect at least *2f* responses before sending out the pre-prepare message and the request. With this approach we guarantee that there are at least *2f + 1* replicas that claim to be able to verify the request. Up to *f* of those replicas may be faulty. However, the *f* other non-faulty replicas that verified the request will send out prepare messages and the non-faulty replicas who were unable to verify the request can use this quorum (including the pre-prepare message) of prepare messages for verification, guaranteeing that all replicas are able to verify the request. Even though this approach has the undesirable effect of adding an extra two rounds of communication even for correct requests, the improvement in performance in the presence of malicious clients is significant since time to complete view changes grows exponentially due to the exponential growth of the request/view-change timeouts.

4.2 Corrupted Primary MAC

In this case, the view change is trigged by the inability of a non-primary replica to determine whether the primary MAC was corrupted. Since all correct replicas forward previously unseen requests to the primary, an obvious way for the primary to verify the validity of the request without us-

ing the MAC would be to collect *f+1* such forward messages. If such a quorum is achieved, then the primary is guaranteed that at least one non-faulty replica received the correct request and verified it using its own MAC.

However, the malicious client may counter this move by sending correct MACs to at most *f* non-faulty replicas. Then all replicas that authenticate the request will forward the request and start request timers. However, the primary does not receive a quorum of forward messages and is unable to process the request. An eventual view change operation will ensue from this scenario.

Once again we use the valid-request messages that contain the digest of the request in question to deal with this situation. When a replica receives a previously unseen request directly from the client, it will first attempt to authenticate the request using its MAC. If it is unable to authenticate the request, it does nothing. If the authentication is successful, the replica will forward the request to the primary, start the request timer, and multicast valid-request messages to the replica group. If the request timer expires, the replica will send out a view-change message only when it has received at least *2f* other valid-request messages. If such a quorum is achieved, a replica can safely conclude that the primary would have been able to verify the request either using its own MAC or a quorum of forwards since at least *f* other non-faulty replicas would have forwarded the request to the primary. Even if the primary was able to verify the request, the request timers may still expire. However, this is desirable since it will force slow primaries to loose their status as primary.

Once the primary is able to verify the request, the first attack case still applies, since with just *f+1* forwards the primary has insufficient information to determine whether *f+1* non-faulty replicas received a valid MAC. However, it cannot wait for *2f* forwards, since fewer than *2f* non-faulty replicas may have received a correct MAC. In such a case the primary may not be able to collet enough valid-request messages and consequently will not send a pre-prepare message to process the request. However, another non-faulty replica may receive *2f* valid-request messages (with the participation of the faulty replicas). When the request timer of this replica times out it will then decide to vote for a view-change. To avoid this scenario, the primary must multicast the digest of the request along with the authenticator when it receives *f+1* forward messages. It must then collect *2f* valid-request messages (possibly including the messages sent during the previous phase) before sending out a pre-prepare message with the new sequence number.

5 Discussion

It has been claimed [6] that view change operations are sufficiently fast with the use of MACs to avoid any serious performance degradation due to these attacks. However, since the attacks can be repeated, and since the exponential growth in timeouts may lead to longer waiting times before starting view change operations, further performance evaluation is needed to assertion the validity of this claim.

While our solution to the first form of attack decreases throughput for in the absence of faulty clients, we believe that it would significantly improve the performance during attacks by malicious clients. Furthermore, we can improve throughput in the normal case by only adding the extra round trip only for requests from clients that previously sent a request that lead to a view change. The extra round of communication required to tolerate the second form of attack runs in parallel with the regular CLBFT algorithm.

6 Conclusion

In this paper we have identified two forms of attacks by malicious clients that significantly impact performance of replica groups that use CLBFT to tolerate Byzantine faults and proposed modifications to guard against each attack. We plan to conduct several experiments with and without our modifications to gauge the performance of replica groups. We also plan to investigate whether reverting back to using public key cryptography to authenticate request messages will result in better overall performance.

References

[1] M. Castro and B. Liskov. Practical Byzantine Fault Tolerance. In *Proc. 3rd Symp. on Operating Systems Design and Implementation*, pages 173–186, 1999.

[2] K. P. Kihlstrom, L. E. Moser, and P. M. Melliar-Smith. The SecureRing group communication system. *ACM Transactions on Information and System Security*, 4(4):371–406, 2001.

[3] P. Narasimhan, K. P. Kihlstrom, L. E. Moser, and P. M. Melliar-Smith. Providing Support for Survivable CORBA Applications with the Immune System. In *Proc. 19th Intl. Conf. on Distributed Computing Systems*, pages 507–516, 1999.

[4] B. Prenel and P. van Oorschot. MDx-MAC and Building Fast MACs from Hash Functions. In *Proc. 15th Conf. on Advances in Cryptology*, pages 1–14, 1995.

[5] R. Rivest. RFC 1321: The MD5 Message-Digest Algorithm, 1992.

[6] Miguel Castro. *Practical Byzantine Fault Tolerance*. PhD thesis, Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, 2001.