Security Issues for Clusters, Clouds , and Data Centres

Anusha Katha
Dept. of Electrical and
Computer Engineering
Queen's University
Kingston, Canada
Email: 22ak3@queensu.ca

***Abstract*—High-Performance Computing (HPC) systems are primarily concerned with enhancing computing performance. It has a competitive processing capacity in terms of memory availability and computation speed. HPC facilities are priceless computational assets that must be properly protected and kept from criminal usage. As a result, vulnerabilities are fundamental problems in HPC systems since most of the resources used or stored are typically sensitive and very profitable data. In this paper, I thoroughly examine the security of HPC systems from the standpoint of Hadoop, Map Reduce, G-Hadoop, RDMA based HPC networks including well-known threats and widely applied defences.**

***Keywords*—** HPC, Hadoop, G-Hadoop, RDMA, Security issues, Vulnerabilities

1. Introduction

High-performance computing, a technology that is widely used in industries like aerospace, automotive, semiconductor design, energy exploration, financial computing, weather forecasting, and nuclear simulation, among others, focuses on how to increase the computing's performance through software development, parallel algorithms, and computer architecture, among other things. It is becoming a crucial method for conducting scientific research. Currently, clusters and grids are the two most popular and different ways to use HPC parallelism in both industry and academics[28].

One reason is that HPC becomes a prominent target of attack when it is a crucial component of a security system, as it was in the airport case. These factors also expose you to additional risks[35]. HPC-powered "data lakes" utilized for big data may not have sufficient security measures in place. These flaws have a variety of causes, but they frequently result from the hasty development of a data repository, maybe from several sources. For instance, data streams from inputs as disparate as weather reporting, financial markets, device logs, geological instrumentation, and so forth may be combined in a study endeavour. Applications and software that run on HPC infrastructures have increased along with the performances and capabilities of HPC infrastructures, particularly with software applications of open-source frameworks. However, attackers find them to be more and more appealing targets[29]. HPC security, which is one of the issues that users and HPC system maintainers are most concerned about, was not a big issue in the past. since the majority of HPC systems were initially created by a small group of devoted users for private usage. Since HPC systems are now frequently shared by several users, especially with the advent of the concept of HPC-as-a-service, it has become one of the key concerns and barriers to their expansion. Given their generally heterogeneous nature, clustered HPC systems may require multiple management systems to operate. This can slow down the implementation of security policies and processes like security patch management with vulnerabilities un-remediated as a result.

Cybersecurity is a key component of High-Performance Computing as it is essential to ensure the protection, availability, confidentiality and authenticity of the data and intellectual property used and generated by those large computing systems. In order to create trustworthy and responsible systems, the security of HPC systems has long been a research priority[30,31]. Authorized users must only use computing power or be kept from authorized users who might otherwise engage in prohibited activities. On the other hand, since most tasks and resources run or stored in HPC systems are typically sensitive and highly profitable data, legal users of computing resources want to protect communications between users and resource owners to be safe and keep their data away from malicious attackers. When hacked, it may have dire repercussions. Other security issues like denial-of-service are also a concern for HPC systems that get numerous user access requests.

There are still conventional security flaws in HPC systems. For instance, most HPC systems are multiuser systems, which necessitate the usage of passwords to authenticate users. However, given the features of HPC systems, conventional security solutions frequently fall short [32]. In most cases, the locations where these high-performance computers are employed have a sizable network with thousands of servers and various kinds of small units under their management. Therefore, maintaining the security of such a big network level is a difficult challenge. Almost everyone with access to a computer has some "high-performance" processing [33]. These new home computers experience all the security issues commonly observed on supercomputers.

The following three elements can be linked to the unique HPC system vulnerabilities connections with high bandwidth High bandwidth connections between the HPC infrastructure and the customers, enabling consistent interaction between users and systems, which have been an essential requirement as the practice of HPC-as-a-service in industry[34]. Hackers may use these high bandwidth connections to perform (i) denial-of-service attacks. (ii) Strong computational capability. On the one side, the enormous processing power can be used by unimportant people to address scientific issues or other difficult computing-intensive problems. On the other side, hostile attackers may exploit it in order to conduct brute-force attacks and other types of unlawful activities. Large storage capacity is (iii). A big storage capacity is a feature of almost all HPC systems. However, hostile attackers are attracted

to large-capacity disc storage because it can be used to store illicit data like pornographic multimedia.

The other sections of this paper are structured as follows: Section 2 gives the background related to Hadoop, G-Hadoop and RDMA. In Section 3, we analyzed the authentication of Hadoop. Section 4 gives you an insight into a security framework for G-Hadoop. In section 5, security issues in RDMA-based HPC are discussed. Section 6 gives you the conclusion.

2. Background

2.1. Hadoop

The data is in multiple formats like emails, images, audio, and videos which come under two main categories which are semi-structured and unstructured data. This data is collectively called Big data[1], and it is impossible to handle this big data by normal processors and storage units. This is where Hadoop comes into the picture. Hadoop stores and processes a vast amount of data efficiently using a cluster of commodity hardware[2].

Modern research and commerce are centred on big data analysis. Online transactions, emails, audio and video files, click streams, logs, posts, search queries, health records, social networking interactions, scientific data, sensors, and mobile devices and their apps all provide this data[3][4].

The open-source Hadoop framework implements the MapReduce algorithm, which underlies Google's method for searching the distributed data sets that make up the internet. The obvious question from this definition is: What are maps, and why do they need to be reduced? Using conventional methods, massive data sets can be quite challenging to examine and query, especially when the queries are quite complex. The mapping of the MapReduce algorithm essentially divides the query and the data set into component pieces. To produce results quickly, the mapped query components can be evaluated simultaneously or reduced.[2]

MapReduce applications built using Hadoop, with a comprehensive examination of the framework's elements, examples of Hadoop in use for a wide range of data processing jobs, and more. MapReduce is a difficult concept to understand conceptually and apply.

Hadoop lacks authentication by default, making it simple to make damaging changes and potentially attack other users or the processing cluster. Hadoop clusters can be configured to use the Kerberos authentication system but doing so requires additional work [5–7]. Similar to this, HDFS's default setup offers a minimum level of protection for the data saved by adhering to Unix access control. However, the HDFS cluster does not by default provide a means for identifying a user or group, even though it is assumed that the user will accurately represent himself.

Undoubtedly, this makes it very simple for an antagonistic client to view and alter data that belongs to other users. Similar to the Hadoop framework, HDFS may be set up to identify users based on their Kerberos credentials[9]. However, the setting necessitates additional work and, as in the case of Hadoop, is probably not carried out in all installations.

2.2. G-Hadoop

A distributed system with numerous clusters, such as a Grid infrastructure[10,11], a Cloud[12,13], distributed virtual machines[14], or a multi-data centre infrastructure[15], is the aim of G-Hadoop[8], a MapReduce implementation. A master/slave communication model underpins the MapReduce architecture of Hadoop, with one Job Tracker serving as the master and several Task Trackers serving as slaves. For user identification and task submission, the current GHadoop system recycles the Hadoop methods built for single-cluster situations. HDFS, the native distributed file system for Hadoop, is replaced with the Gfarm file system[16] by G-Hadoop. Users can submit their MapReduce applications to G-Hadoop, which executes map and reduce tasks across multiple clusters. With the widely used MapReduce model, G-Hadoop offers a parallel processing environment for large data sets across dispersed clusters. It implements fine-grained data processing parallelism and achieves high throughput data processing performance compared to data-intensive workflow systems. Additionally, G-Hadoop can offer fault tolerance for large-scale big data processing by replicating the map and reducing jobs[8].

2.3. RDMA

Modern systems have increasingly started to use remote direct memory access (RDMA) primitives as a communication method that ensures high performance and efficient resource use in recent years. Lower latency, more bandwidth, and less CPU consumption are all benefits of RDMA. In data centre applications, RDMA technologies like InfiniBand (IB)[17] or RDMA over Converged Ethernet (RoCE)[18] are growing more popular and are also gaining popularity in cloud contexts. There are several RDMA designs, however, performance, not security, is the major consideration when designing RDMA systems. As a result, RDMA architectural security is essential, although the possible security effects of RDMA communication are mainly unexplored.

Multiple plaintext access tokens are used by current RDMA technologies to enforce isolation and prevent unwanted access to system memory[19]. Any entity that acquires or guesses these tokens can read and write memory regions made accessible by using RDMA on any computer in the network, jeopardizing not only the secrecy but also the integrity of the applications. This is because the tokens are transmitted in plaintext. RDMA systems rely on isolation and the presumption that the underlying network is a well-protected resource to prevent the compromise of these access tokens. Otherwise, a malicious switch or a wire that is bugged can allow an attacker to listen in on access tokens for bypassing packets that are travelling between two connecting parties. Unfortunately, the present RDMA specifications do not provide encryption and authentication of RDMA packets (such as those suggested by Taranov et al. [20]).

3. Authentication in Hadoop

Delegation tokens, block access tokens, and job tokens are three unique sorts of tokens that are used as part of an authentication system for Hadoop [22,23]. The interaction between a user and HDFS is divided into two parts: (i) a user connects to a NameNode using the Hadoop RPC libraries, and all RPCs connect using the Simple Authentication and Security Layer, which uses Kerberos, DIGEST-MD5, or a delegation token; and (ii) a user connects to a DataNode using a streaming socket connection, which is secured using a block access token. The three different token kinds operate as follows:

- **Delegation token.** Between a user and NameNode, a delegation token serves as a private key. After authenticating a user, NameNode creates a delegation token using Kerberos, then uses the token to authenticate the user again without using Kerberos

- **Block access token.** Block access tokens are issued by NameNodes utilizing symmetric-key schemes, where

NameNode and all of the DataNodes share a secret key, to offer authentication policies to DataNodes through the transfer of permission information from NameNode to DataNodes . A user asks NameNode for the block ids and locations of a file when they wish to access it in HDFS. If NameNode can confirm the user's authority and validity, it will send block ids, locations, and a block access token in response. The associated DataNode receives the block id from the user together with the block access token, and before granting access to that block, the DataNode verifies the block access token.

- **Job token.** When a MapReduce job is submitted, JobTracker creates a job token in the form of a secret key. Each job's job token is stored by JobTraker as a component of the job and distributed to TaskTrackers. At TaskTrackers, tasks are authenticated using the job token. Since the MapReduce model itself is security-neutral, the early implementation of Hadoop lacked a built-in security mechanism. However, as MapReduce in general and Hadoop in particular became more widely used, the need for security became more urgent. This prompted a number of recent efforts, which we will examine next, to attempt to add various security features to Hadoop.

3.1. Token based Authentication

There are some proposed authentication techniques for Hadoop, One of them is Token-based authentication for the Hadoop platform[21]. The paper suggests a brand-new authentication technique that is both safe and scalable. The authentication protocol makes it possible for new clients to sign up for Hadoop and makes use of a token mechanism to scale the authentication process. They have integrated HTBA and other Hadoop authentication mechanisms in the Deter lab environment. Security projects are tested in the Deter lab. The performance comparison of the authentication protocols is done in the paper using the WireShark tool and the Crypto++ benchmark. The SVO logic belief and the AVISPA verification tool are used in a formal security analysis to demonstrate the HTBA's security. A security feature table that contrasts HTBA and the Hadoop authentication procedures is the last step.

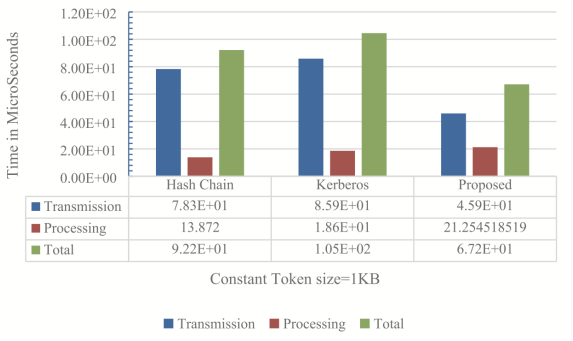

Figure 1: Transmission delay, Processing delay, Total comparison for the three protocols.

The suggested HTBA protocol is contrasted in the study with both the standard Kerberos protocol and a related mechanism known as Hash-Chain. They used SVO logic to assess the three protocols' accuracy and security guarantees and discovered that

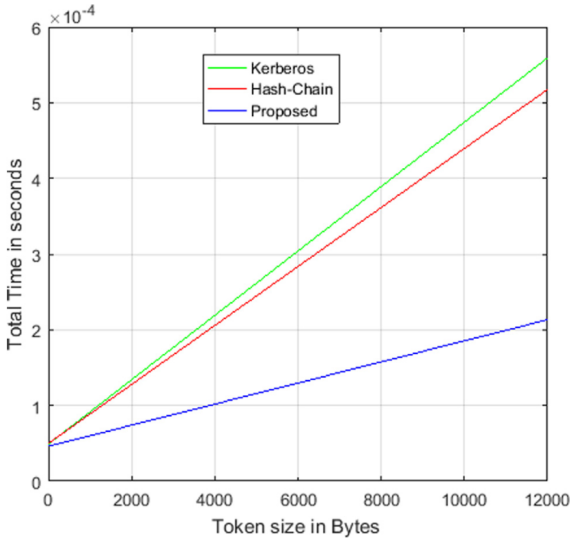

Figure 2: Transmission delay versus Token size.

Kerberos has only one vulnerability related to weak passwords. The Hash-Chain protocol adds significant transmission time, has many redundant processes, and is vulnerable to replay attacks. The HTBA protocol completely eliminates the weaknesses of the other two protocols. Additionally, they assessed how well the three protocols performed in terms of processing and transmission delay(Figure 1). The outcome demonstrates how Kerberos and Hash Chain protocol performance is impacted by token size. The HTBA protocol is better suited for scalability since it can cut the duration to 25% at a token size of 20 KB and has a lower transmission delay than the other two when the token size is greater than 2 kB(Figure 2).

4. Security Framework for G-Hadoop

The SSL protocol[24], the Globus Security Infrastructure, and Java security solutions are examples of effective security measures. Secure communication over open networks is made possible by the Secure Sockets Layer (SSL) cryptographic protocols and its successor Transport Layer Security (TLS). In G-Hadoop, each cluster must have its SSH (Secure Shell) connection established between the user and it. A security framework in G-Hadoop is proposed for big data computing across remote Cloud data centres[25]. Users will have less work to do because there will be one sign-on process, which only needs a username and password.

While utilizing SSL for communication between the master node and the CA server, this security model adheres to the GSI's authentication paradigm. Jobs can then be sent to the clusters without requesting any additional resources by just logging on to the master node or by entering the user's username and password. The security framework uses a mechanism that mimics the SSL handshaking phase[26] to create a secure connection between the master node and slave nodes. The following are some terminology used:

- **User Instance:** A user instance is an object that is created by the master node for a user who has active jobs – the job is initialized or in execution.
- **Proxy credential and slave credential:** The proposed security system defines two types of certificates as slave credentials and proxy credentials. They have the same CA

stamp on them (Certification Authority). Proxy credentials are thus created for single-use and have a brief lifespan in order to safeguard the authentication process from MITM and replay attacks. The identity of the CA, the time until it expires, the master node's public key as well as its life cycle, and a randomly generated message are all included in a proxy credential.A slave credential is owned by a slave node and has a longer life cycle than a proxy credential. When a communication channel is being established between a master node and a slave node, the master node uses slave credentials to authenticate the slave nodes.

- **User sessions**The interactive information exchange between two communication parties is commonly referred to as a session.

4.1. The Security Model

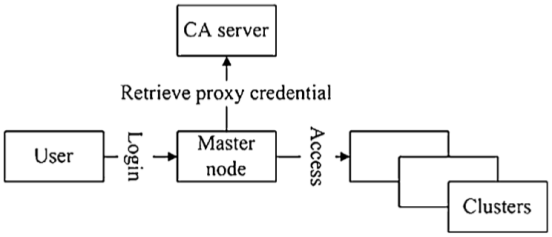

Figure 3: The G-Hadoop security architecture.

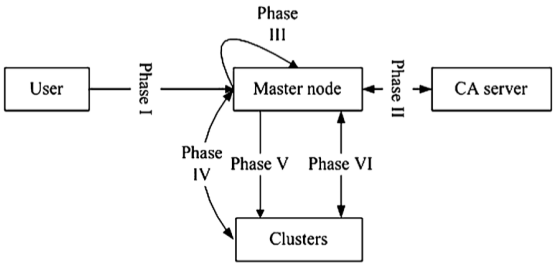

Figure 4: The work flow of the authentication procedure.

The proposed security framework's architecture model is depicted in Fig. 3. The newly created security framework maintains the master-slave architecture of the current G-Hadoop system while extending it. The CA server, which is required to provide proxy credentials as well as slave credentials, is an additional component in this architecture.

The authentication process and the interplay of the security framework's design elements are shown in Fig. 3. The following stages make up the entire work flow: user authentication, proxy credential assignment, master node preparation for authentication, master node and slave node authentication, task execution, termination, and disconnection.

- **Phase I: User authentication**
  A user does not required to log in to each slave node when using the security model. Instead, a single login using his user name and password combination, U, P, to the master node is sufficient. The master node checks its database, which maintains the user authentication information, after receiving the user authentication information (U,P), to see if the user name is present, if the password is accurate, and if the user is authorized to utilize G-Hadoop resources. The master node provides notification to the user in the event that one of these tests fails. If not, the master node authenticates the user.
- **Phase II: Applying and issuing a proxy credential**
  The first stage, Phase II(a) in Fig. 4, is to establish a secure connection between the master node and the CA server in order to apply and issue a proxy credential. In this stage, the SSL protocol [9] is used to offer the secure connection. As shown in Phase II(b) of Fig. 4, the master node then requests to apply a proxy credential for the user instance. A proxy credential must be unique for every use case in order to prevent abuse, even when used for the same user instance but different jobs. A proxy credential is the authentication data for a user instance on the slave nodes. The master node creates a random key pair called MN Pub, MN Prv for this reason.
  Later, when the master node and slave nodes are being authenticated, this key pair is used. The master node transfers the public key MN_Pub to the CA server while keeping the private key MN_Prv. The CA server generates a random message CA_Rand and specifies the life cycle for the master node's randomly generated public key MN_Pub after receiving it from the master node. The life cycle of the related user session, which will be created in the following step, is controlled by the life cycle of MN_Pub as well. The CA server then uses the method of digital signature to sign a proxy credential for the current request of the master node. The master node's public key MN_Pub, its life cycle, the CA_Rand message, the identity of the CA, and its expiration date are all included in the proxy credential. The newly produced proxy credential and the random message CA_Rand are then transmitted to the master node.
- **Phase III: Generating user session and preparing the authentication for slave nodes**
  For eventual communication and interaction with the slave nodes, the master node creates a user session of the current user instance during this stage. To ensure that the user session is unique and to shield the user's privacy from the slave nodes, the user session U_Session only employs the username's MD5 hash value and a random message.
  U _Session = MD5(U + CA_Rand)
  The CA server specifies the life cycle of the public key MN_Pub, which is signed in the proxy credential, and the user session follows suit.
  MN_Rand = PBA.encrypt(CA_Rand, MN_Prv)
  MN_U _Session = PBA.encrypt(U _Session, MN_Prv)
- **Phase IV: Handshaking between the master node and slave nodes**
  This stage is comparable to the SSL connection's handshaking stage. This stage involves setting up a secure communication channel for future data transmission and requiring mutual authentication between the master node and slave nodes. All slave nodes do the identical operations during this phase. The messages that were prepared in the last step are sent to the slave node by the master node. The proxy credential and the authentication data MN Rand from the master node are used by the slave node to attempt to authenticate the master node. With the help of the CA server's public key for the digital signature, it first decrypts the proxy credential. CA_Rand = PBAdecrypt(MN_Rand, MN_Pub) If the master node is

verified, the slave node provides feedback to the master node by sending a confirmation message, which it then decrypts to obtain the user session data. MN_U_Session using the MN_Pub public key
U _Session = PBA.decrypt(MN_Rand, MN_Pub)
To obtain the encrypted message SN_Rand, the slave node uses its own private key SN_Prv to encrypt the random message from the CA server:
SN_Rand = PBA.encrypt(CA_Rand, SN_Prv) The slave node's private key SN_Prv is also used to encrypt the user session U_ Session. Encrypting the user session serves to ensure that the user session sent by the master node and received by the slave node are identical.

- **Phase V: Job execution**
  The G-Hadoop system runs the user-submitted jobs during this stage. The job is carried out in a manner akin to G-Hadoop without the intended security framework.
- **Phase VI: Terminating a job**
  When all of the tasks associated with the current job have been completed on the slave nodes or a fatal error has occurred during execution, which prevents further task execution, the active job is terminated. The user receives the outcome or some information on the fatal mistake from the master node. Along with the user session, it sends a job-finished message to each of the participating slave nodes. The latter deletes all user session-related data, including the proxy authentication information and the encryption data SCA,SC Key. The slave node ends the job's current task if a fatal error resulted in the job-finished message.

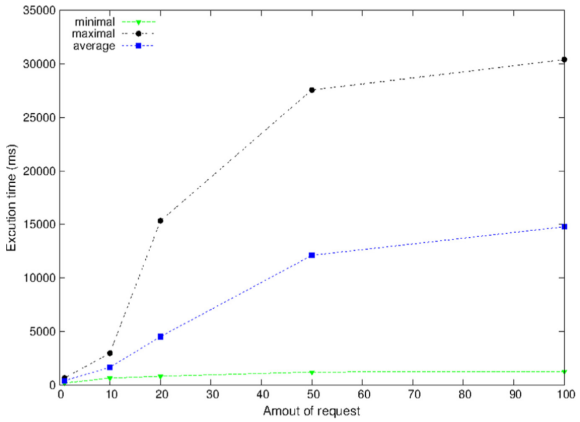

Figure 5: Execution time for requesting proxy credentials on the master node.

4.2. Analysis

The quantity of requests below the threshold of 50 has a significant impact on how quickly proxy credentials are applied and distributed as shown in Fig. 5. The influence decreases as the number of requests increases beyond 50. Therefore, a sufficient value should be chosen for the threshold for the maximum number of application requests. The effectiveness of the security architecture is constrained if it is too tiny.

These security features enable the security architecture to safeguard against the most frequent attacks, including MITM, replay, and delay attacks, and provide secure communication of Hadoop across open networks. Additionally, it uses a variety of safeguards to prevent misuse or abuse of G-resources. Hadoop's Overall, it offers a reliable and comprehensive single-sign-on process solution for the user to access Hadoop.

5. RDMA based HPC

ReDMArk[27] demonstrates that the IB-based architectures' present security measures are insufficient to thwart both in-network and end-host attackers, compromising the secrecy and integrity of RDMA applications. The paper demonstrated numerous flaws in the implementation of RDMA-capable network interface cards (RNICs) and the design of IB-based architectures. It used these flaws to enable strong attacks like packet injection using impersonation and unauthorized memory access, and denial-of-service (DoS) attacks.

In the proposed adversary model, three parties were taken into account (see Figure 6): an RDMA service that hosts one or more RDMA applications, a client that communicates with the service using RDMA, and an adversary who is able to connect to the RDMA service legally but tries to circumvent the security measures in place.

5.1. Adversary Model

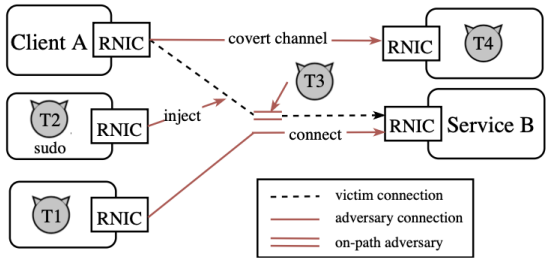

Figure 6: Illustration of the adversary model including potential adversary locations.

Model T1 First an adversary that is located at a different end host than the victim (off-path) and have rightfully obtained these hosts (e.g., by renting an instance in a public cloud) is considered.

Model T2 Second, the attackers who can actively compromise end hosts, manufacture communications, and inject messages are taken into account (perhaps off-path). The adversary needs root administrative access to insert a random RDMA request successfully.

Model T3 Third, network-based attackers who are situated between the victim and the service are taken into account. In on-path assaults, the attacker must have access to routers or links that connect the victims (e.g., rogue cloud provider, rogue administrator, malicious bump-in-the-wire device).

Model T4 Finally, an opponent that employs RDMA as a stealthy channel for data espionage is taken into consideration. To accomplish this, the adversary modifies the code or libraries that the victim executes (for example, by employing malware) such that it creates an RDMA connection to an attacker machine that is capable of RDMA in the same network as the victim (e.g., by renting an instance in a public cluster).

The analysis set up includes multiple IB-based architectures such as native InfiniBand (IBA), RoCEv1, and RoCEv2. Some of the vulnerabilities are discussed below.

5.2. Vulnerabilities

V1 Memory Protection Key Randomness IBA requires that RDMA read/write requests include a remote memory access key (rkey), which is negotiated between communicating peers and verified at the remote RNIC, to secure remote memory against unauthorized memory access. A connection error that results in disconnection is brought on by packets with an invalid rkey.Analysis is done on the randomness of the rkey generation process for various RNIC models and drivers. Rkey generation is independent of the buffer's address and size that must be registered for all tested devices. The production of a rkey is unaffected by changes to access flags. Only prior registration and deregistration operations are taken into account when generating rkeys. Attackers would typically be successful with one out of every three guesses if they predicted that the difference with the highest likelihood would increment the following key. An attacker can highly likely estimate the rkey of upcoming registrations using the 2.85 bits of entropy produced by the conditional entropy computation of the mlx5 algorithm.

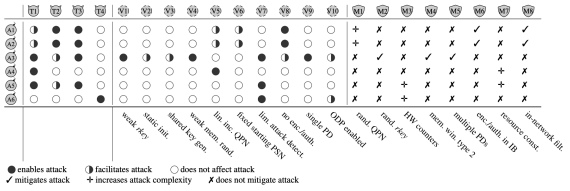

Figure 7: Overview of dependency on attacker location and vulnerabilities for attacks on RDMA combined with an overview of mitigation mechanisms that thwart the attack.

V2 Static Initialization State for Key Generation The RNICs based on the bnxt_re and mlx4 drivers are initialized using static state in addition to the limited amount of rkeys, and the same set of keys remains across various physical reboots of the machine. The same keys will continue to be used even after the physical computer has restarted, assuming an adversary has observed the whole key set.

V5 Linearly Increasing QP Numbers All of the drivers and tested devices have consecutive QP numbers assigned to them. Predicting prior or subsequent QP numbers is simple, if an adversary registers a QP himself or witnesses a QP registration request. Additionally, because IBA employs 24 bit QP numbers, an RNIC can only support 2^{24} QP connections at a time.

V6 Fixed Starting Packet Sequence Number (PSN) The RDMA implementation provides two methods for creating connections: either via the native RDMA connection interface or the RDMA connection manager [29].The connection settings, such as the destination QP number and the local and distant starting PSNs, are defined by the application developers via the native connection interface. The RDMA connection manager relieves application developers of this responsibility by generating a beginning PSN at random (using a cryptographic pseudorandom number generator), enabling the setup of RDMA connections comparable to TCP sockets. Many open-source RDMA apps choose to use the native interface and configure the initial PSNs by hand. Predicting the starting PSNs of established connections becomes significantly easier if the starting PSNs are not randomized for each QP connection.

V8 Missing Encryption and Authentication in RDMA Protocols The header and payload of RDMA packets are not authenticated nor encrypted by the RDMA network protocols in use today. A malicious party can change any byte in an RDMA message's packet payload or spoof any field in the packet header. Recalculating packet checksums—whose algorithms and seeds are understood and described by the IBA—is the only step necessary for in-network packet modification.

5.3. Mitigation Methods

The paper also proposes some mitigation technique to overcome the vulnerabilities(see Figure 7).

M1 Randomization of QPNs A pool of disconnected QP descriptors with random QPNs is created by an RDMA application. One of the QP descriptors is obtained and registered as soon as a QP connection is necessary. The RDMA host will experience some overhead as a result of this step, but it can be implemented without changing the RDMA protocols as is, bringing the total number of packets an attacker must insert up to 224. Such bruteforce efforts could be discovered since contemporary RNICs include hardware counters that are accessible by apps.

M3 Hardware Counters in RNICs On the basis of the mlx5 driver, recent RNICs from Mellanox support port and hardware counters that are available to RDMA applications [26]. These counters make it possible to precisely monitor debugging, load estimating, and error detecting requests. For instance, access errors caused by invalid requests may be tracked using resp_remote_access_errors. With the aid of these counters, malicious RDMA traffic flooding attacks could be recognized.

M7 Per-Client Resource Constraints The number of continuously open QP connections and the amount of resources allotted to each client should be restricted by RDMA-capable devices. Otherwise, resource exhaustion-based attacks cannot be prevented in advance. With per-client resource limitations in place, a resource exhaustion assault would require cooperation from a large number of end points in order to be successful. InfiniBand adapter identities for native IB connections and IP addresses for RoCE connections might both be used to allocate resources per-client.

The implementation of IB-capable NICs and the design of IBA both have numerous problems and weaknesses, as demonstrated by ReDMArk. These flaws make it possible for an attacker to inject packets, acquire unauthorised access to other clients using an RDMA-based service, perhaps with disastrous results, and effectively stop communication in RDMA networks. Weak RDMA security causes real-world vulnerabilities in RDMA-enabled systems because InfiniBand is deployed in public infrastructure and more providers intend to embrace RDMA networking. The risks posed by RDMA networking must be understood by those who develop RDMA-enabled systems, and mitigating strategies such as employing type 2 memory windows, a different PD for each connection, and the recommended algorithms to randomize the QPN and the rkey generation should be used.

6. Conclusion

In this paper, we have examined authentication in Hadoops, security concerns with MapReduce, G-Hadoops, and RDMA-based HPC networks. HPC is the integration of computational resources used as a single resource. There are numerous potential attack vectors and security challenges. Since HPC is constantly evolving, we have a new architecture every day, and every new architecture has its own set of security issues. Technology advancement raises more questions about security. Numerous publications attempted to break into or attack HPC systems, and they were successful, demonstrating how much room there is for security improvements in HPC. Although performance is often the main focus of architectures, security considerations should also be given attention. There

is no purpose for an architecture if its performance is excellent but its security is poor.

References

[1] Sagiroglu, S., Sinanc, D. (2013, May). Big data: A review. In 2013 international conference on collaboration technologies and systems (CTS) (pp. 42-47). IEEE.

[2] Lam, C. (2010). Hadoop in action. Simon and Schuster.

[3] Zikopoulos, P., Eaton, C. (2011). Understanding big data: Analytics for enterprise class hadoop and streaming data. McGraw-Hill Osborne Media.

[4] Schneider, R. D. (2012). Hadoop for Dummies Special Edition. John WileySons Canada, 978-1.

[5] http://web.mit.edu/kerberos/.

[6] O. O'Malley, K. Zhang, S. Radia, R. Marti, C. Harrell, Hadoop security design. Tech. Rep, Yahoo, Inc., 2009.

[7] D. Das, O. O'Malley, S. Radia, K. Zhang, Adding security to Apache Hadoop. Hortonworks Technical Report 1, 2010.

[8] Wang, L., Tao, J., Ranjan, R., Marten, H., Streit, A., Chen, J., Chen, D. (2013). G-Hadoop: MapReduce across distributed data centers for data-intensive computing. Future Generation Computer Systems, 29(3), 739-750.

[9] Neuman, C., Yu, T., Hartman, S., Raeburn, K. (2005). The Kerberos network authentication service (V5) (No. rfc4120).

[10] Kołodziej, J., Xhafa, F. (2012). Integration of task abortion and security requirements in GA-based meta-heuristics for independent batch grid scheduling. Computers Mathematics with Applications, 63(2), 350-364.

[11] Ranjan, R., Harwood, A., Buyya, R. (2012). Coordinated load management in peer-to-peer coupled federated grid systems. The Journal of Supercomputing, 61(2), 292-316.

[12] Ranjan, R., Mitra, K., Georgakopoulos, D. (2013). MediaWise cloud content orchestrator. Journal of Internet Services and Applications, 4(1), 1-14.

[13] Wang, L., Kunze, M., Tao, J., von Laszewski, G. (2011). Towards building a cloud for scientific applications. Advances in Engineering software, 42(9), 714-722.

[14] Wang, L., Chen, D., Zhao, J., Tao, J. (2012). Resource management of distributed virtual machines. International Journal of Ad Hoc and Ubiquitous Computing, 10(2), 96-111.

[15] Zhang, W., Wang, L., Liu, D., Song, W., Ma, Y., Liu, P., Chen, D. (2013). Towards building a multi-datacenter infrastructure for massive remote sensing image processing. Concurrency and Computation: Practice and Experience, 25(12), 1798-1812.

[16] Tatebe, O., Hiraga, K., Soda, N. (2010). Gfarm grid file system. New Generation Computing, 28(3), 257-275.

[17] Pfister, G. F. (2001). An introduction to the infiniband architecture. High performance mass storage and parallel I/O, 42(617-632), 102.

[18] Subramoni, H., Lai, P., Luo, M., Panda, D. K. (2009, August). Rdma over ethernet—a preliminary study. In 2009 IEEE International Conference on Cluster Computing and Workshops (pp. 1-9). IEEE.

[19] Rothenberger, B., Taranov, K., Perrig, A., Hoefler, T. (2021). ReDMArk: Bypassing RDMA Security Mechanisms. In 30th USENIX Security Symposium (USENIX Security 21) (pp. 4277-4292).

[20] Taranov, K., Rothenberger, B., Perrig, A., Hoefler, T. (2020). sRDMA–Efficient NIC-based Authentication and Encryption for Remote Direct Memory Access. In 2020 USENIX Annual Technical Conference (USENIX ATC 20) (pp. 691-704).

[21] Haggag, M., Tantawy, M. M., El-Soudani, M. M. (2022). Token-based authentication for the Hadoop platform. Ain Shams Engineering Journal, 101921.

[22] O. O'Malley, K. Zhang, S. Radia, R. Marti, C. Harrell, Hadoop security design. Tech. Rep, Yahoo, Inc., 2009.

[23] D. Das, O. O'Malley, S. Radia, K. Zhang, Adding security to Apache Hadoop. Hortonworks Technical Report 1, 2010.

[24] Rescorla, E. (2002). SSL and TLS designing and building secure systems.

[25] Zhao, J., Wang, L., Tao, J., Chen, J., Sun, W., Ranjan, R., ... Georgakopoulos, D. (2014). A security framework in G-Hadoop for big data computing across distributed Cloud data centres. Journal of Computer and System Sciences, 80(5), 994-1007.

[26] Hickman, K., Elgamal, T. (1995). The SSL protocol.

[27] Rothenberger, B., Taranov, K., Perrig, A., Hoefler, T. (2021). ReDMArk: Bypassing RDMA Security Mechanisms. In 30th USENIX Security Symposium (USENIX Security 21) (pp. 4277-4292).

[28] Pellerin, D., Ballantyne, D., Boeglin, A. (2015). An introduction to high performance computing on aws. Amazon Whitepaper.

[29] Peisert, S. (2017). Security in high-performance computing environments. Communications of the ACM, 60(9), 72-80.

[30] Foster, I., Karonis, N. T., Kesselman, C., Tuecke, S. (1998). Managing security in high-performance distributed computations. Cluster Computing, 1, 95-107.

[31] Blanc, M., Briffaut, J., Coullet, T., Fonda, M., Toinard, C. (2010, July). Protection of a shared hpc cluster. In 2010 Fourth International Conference on Emerging Security Information, Systems and Technologies (pp. 273-279). IEEE.

[32] Apostal, D., Foerster, K., Chatterjee, A., Desell, T. (2012, December). Password recovery using MPI and CUDA. In 2012 19th International Conference on High Performance Computing (pp. 1-9). IEEE.

[33] Dowd, Kevin, and Charles Severance. "High-performance computing." (2010).

[34] Luo, Z., Qu, Z., Nguyen, T. T., Zeng, H., Lu, Z. (2019). Security of HPC systems: From a log-analyzing perspective. EAI Endorsed Transactions on Security and Safety, 6(21), e5-e5.

[35] https://www.hpcwire.com/2017/10/17/security-meets-high-performance-computing/