Andrew Gross
COMP 116
Ming Chow
12/13/19

Risks of Rapid Database Deployment: The Misconfigured MongoDB

Abstract

One of the most harmful aspects of the tech industry's rapid growth is just that — its pace. Today, software deployment cycles are completed rapidly and are getting even faster thanks to advances in development and operations theory, such as DevOps. With such pace, however, software development is rife with missteps in management, bugs, and data disasters. For this paper, I will be analyzing how MongoDB's nature as a quick-to-deploy database is a prime suspect in many of the recent data breaches. I will focus on MongoDB specifically by both evaluating its security features and by demonstrating how hasty deployment can be the root cause of a ransomware nightmare. For my evaluation, I will discuss several recent MongoDB breaches as case studies and determine if a pattern among such breaches is a lack of proper configuration.

Introduction

No matter where you are in the tech industry, one thing is for certain. Big tech companies (i.e. Facebook, Google, Microsoft, Amazon, etc.) are focused on developing a wide variety of online services that scale to millions of users. Their goal is to make a product that becomes ubiquitous to modern life. In order to accomplish this, big tech companies work under a set of practices that standardize and safeguard the development and operations of such large software

services. These practices are collectively known as DevOps. Working under the philosophy of DevOps, companies seek to continuously integrate software updates and to provide continuous deployment of software to the end user.[2] Companies working under DevOps intend to gain a competitive edge in the market due to the speed of development and safety of deployment.

When developing software, data that goes in and out of a given service usually produces data that needs to be stored. To keep such data, companies working under DevOps incorporate database deployment into their deployment pipelines. Usually, this involves spinning up a database server via a cloud computing service such as AWS or Google Cloud. DevOps advocates performing these deployment operations programmatically, or rather, via automation.[2] Enter MongoDB; an open-source database that has gained lots of attention from the tech community in recent years and has achieved a status of many companies' goto database for their production servers.[3] Here lies a potential culprit for several of the high-profile data breaches in recent years: a company production server running a MongoDB instance full of user data and, both unfortunately and shockingly, open to the public internet.

To the Public

With an increase in the adoption of DevOps workflows among tech companies,[4] we should take care to consider the risks of companies quickly deploying MongoDB servers to production. In recent years, several large scale data breaches have given MongoDB a negative reputation regarding its security as a database server. Of the following high-profile cases where a MongoDB instance exposes millions of records, the common theme is that the database server is

found deployed unconfigured, without a password, and accepting of all TCP connections via port 27017. This base level of security configuration, or lack thereof, puts the onus of figuring out how to protect one's data immediately on the user as there is no basic security requirement for MongoDB server deployment. In light of the deployment pace advocated through DevOps, it is not surprising that we have seen some horrific data leaks based almost solely on the lack of security enabled on the MongoDB server.

Recent Cases

On February 25th, 2019, security journalist Bob Diachenko at SecurityDiscovery.com discovered one of the largest unprotected MongoDB database to date[5]. The server contained 150GB of user data including email and other personally identifiable information (PII). The data leak totaled over eight hundred million records from the MongoDB instance administered by a company called Verifications.io, providers of an email validation service. Verifications.io deployed this MongoDB production server to the public internet without a password. This example shows, without a doubt, that there exists an issue in DevOps culture where the focus on developing digital infrastructure quickly overshadows one's ability to identify obvious security risks.

In December 2016, CloudPets, a children's toy company, exposed PII of over eight hundred thousand users.[6] The data included over two million voice recordings of children and parents talking to each other through the toy. What might be the cause of such a massive data leak? A public-facing MongoDB server with no authentication configuration in place. CloudPets

had set up their production and development databases on the same public box with no security. Troy Hunt, a security blogger and an Australian Microsoft Regional Director, details the CloudPets data disaster on his blog and connects this instance to slew of similar breaches involving misconfigured public-facing MongoDB servers full of production data.

In 2017, the Kromtech Security Center discovered a public-facing unsecured MongoDB database with thirty-one million client records.[7] This database belonged to an Israeli start-up called Ai.Type, a maker of personalized keyboards for Android and iOS devices. Ai.Type's keyboards require full access to a user's phone data, including data that the app should not have access to such as the owner's full name, their SMS number, email present on the phone, and even geographical location detail. Kromtech researchers and an unknown number of individuals were able to access this data for each user of Ai.Type's app simply by connecting to the public-facing Ai.Type MongoDB instance using a simple Mongo client. Evidently, Ai.Type mishandled not only its collection of user data by being too intrusive with its mobile app's permissions, but Ai.Type also irresponsibly overlooked the obvious security concerns by deploying their unprotected production database server to the public-internet and not to an internal network. Cases like these demonstrate that many companies continue to quickly develop and deploy products without properly configuring such services for production.

Kromtech's Honeypot Experiment

To further the discussion regarding the vulnerable nature of the misconfigured MongoDB server, it is worth discussing Kromtech's 2018 honeypot experiment.[8] Their "honeypot," a type

of service deployed to look vulnerable and with the intention of logging nefarious activity conducted against it,[9] consisted of Splunk deployed on an AWS EC2 instance running an instance of MongoDB version 2.6.10 with two databases full of mock data and no authentication configuration. Kromtech chose to use MongoDB version 2.6.10 in order to imitate many of the MongoDB instances that are still public today. With the Splunk log forwarding set up and the MongoDB accepting by default of all TCP connections on port 27017, Kromtech researchers launched the server to the public internet. Within eleven days, the database server was compromised by a machine with a Chinese IP address. The whole attack, thirteen seconds in duration, consisted of a dropping of the databases, a deletion of the MongoDB Journals (database logs), and the creation of a Readme collection in which the attacker left a ransom note. Evidently, automated scripts actively search for public-facing MongoDB servers and attempt to make connections. With scripts like these automatically scanning the internet for any open database, we should not act surprised when we hear about events like the aforementioned MongoDB server leaks. According to Shodan, in 2018, there were almost fifty-four thousand unsecured MongoDB databases available on the internet.

Action Items

The tech industry must handle user data with more care. It is as simple as that. MongoDB database servers can be as secure as its competitors if those deploying provide the necessary attention and care. Having written code that interfaces with MongoDB security, and directly with the wire protocol, I can say that users can configure for user authentication with a variety of mechanisms including SCRAM, x.509, and Kerberos to name a few. Users can also enable

role-based authorization for server users and groups and TLS/SSL for traffic encryption. Knowing that these features exist, it is a shame that thousands of unsecured MongoDB servers remain open and available on the internet. That said, MongoDB, Inc must do its best to reduce the onus of users to secure their data by developing deployment security requirements that can only be overridden operators with intentionality and care. Users often choose to use MongoDB as a production database for its quick connectivity and scalability.[10] These attributes make MongoDB prime for DevOps's practices of continuous delivery, developing microservices, and rapid deployment.[2] Thus, operators of MongoDB must reduce the pace of their deployments or automate deployment with more caution because the risks of violating user privacy and associated data laws have never been higher.

Conclusion

As DevOps continues to make strides pushing the tech industry towards an increasingly automated world of work, the risk of human error does not necessarily decrease. Scripts written to reduce human input contain human error and that is unavoidable. This goes to show that understanding and care go a long way with regard to how we properly engineer our world. In most cases, the engineering and procedures of working with our buildings, bridges, or even bodies have standards. After exploring this topic of insecure MongoDB server deployment, the tech industry clearly does not have a mature understanding of user safety. If it did, perhaps there would not be tens of thousands of collections of user data waiting to be exploited.

References

[1] Sauce Labs. "Extent of Devops Adoption by Software Developers Worldwide in 2017 and 2018." Statista, Statista Inc., 28 Feb 2018, https://www.statista.com/statistics/673505/worldwide-software-development-survey-devops-adoption/

[2] Freeman, Emily. "What Is DevOps?" *Amazon*, Amazon Web Services, Inc., 2019, https://aws.amazon.com/devops/what-is-devops/.

[3] DB-Engines. "Ranking of The Most Popular Database Management Systems Worldwide, as of September 2019*." Statista, Statista Inc., 3 Sep 2019, https://www.statista.com/statistics/809750/worldwide-popularity-ranking-database-management-systems/

[4] Sauce Labs. "Adoption of Agile Development and Continuous Integration (Ci) in Software Development Worldwide from 2015 to 2018." Statista, Statista Inc., 28 Feb 2018, https://www.statista.com/statistics/673786/worldwide-software-development-survey-agile-development-continuous-integration-adoption/

[5] Diachenko, Bob. "800 Million Emails Leaked Online by Email Verification Service." *Security Discovery*, Security Discovery Consulting, 10 Mar. 2019, https://securitydiscovery.com/800-million-emails-leaked-online-by-email-verification-service/.

[6] Troy Hunt. "Data from Connected CloudPets Teddy Bears Leaked and Ransomed, Exposing Kids' Voice Messages." *Troy Hunt*, Ghost, 20 Dec. 2017, https://www.troyhunt.com/data-from-connected-cloudpets-teddy-bears-leaked-and-ranso

med-exposing-kids-voice-messages/.

[7] Center, Security. "Virtual Keyboard Developer Leaked 31 Million of Client Records." *Kromtech*, Kromtech Alliance Corp. , 5 Dec. 2017, https://kromtech.com/blog/security-center/virtual-keyboard-developer-leaked-31-million-of-client-records.

[8] Center, Security. "How Long Does It Take for a MongoDB to Be Compromised." *How Long Does It Take for a MongoDB to Be Compromised*, Kromtech Alliance Corp., 22 Mar. 2018, https://kromtech.com/blog/security-center/how-long-does-it-take-for-a-mongodb-to-be-compromised.

[9] NortonOnline. "What Is a Honeypot? How It Can Lure Cyberattackers." *Norton*, NortonLifeLock Inc., 2019, https://us.norton.com/internetsecurity-iot-what-is-a-honeypot.html.

[10] "MongoDB at Scale." *MongoDB*, https://www.mongodb.com/mongodb-scale.