Kubernetes, IoT and the Automation of Penetration Testing

Fabian van Dongen
University of Twente
P.O. Box 217, 7500AE Enschede
The Netherlands
f.h.j.vandongen@student.utwente.nl

***Abstract*— The increasing use of Internet of Things (IoT) systems in various industries has raised concerns about their security. One specific area of concern is the security of IoT systems with deployed container orchestrators. Kubernetes, an innovative container orchestrator, is one of the most efficient ways for automating software deployment, scaling, and management. Overall, Kubernetes is much more efficient than other container orchestrators, but this has not been implemented in IoT due to its computational requirements. As lightweight version of Kubernetes K3s has been released which might be implemented in IoT soon. Whether K3s will be a feasible option to use in IoT and whether its security can be tested in the form of automatic penetration testing, is explored in this paper. To do this, a simple IoT system with K3s deployed has been set up and penetration testing has been performed on the system. The system was able to handle K3s, but it has some drawbacks that need attention. The penetration testing worked manually as well as automatically with its restrictions. To implement this in various diverse IoT systems, a lot of variables have to be taken into account.**

***Index Terms*—Kubernetes, K3s, Penetration Testing, IoT, Security**

I. Introduction

IoT is all around you. The smart speakers in your living room, your smartwatch, that car that opens when you are near it, and even your wireless doorbell. There is an incredibly big number of examples of IoT systems. These systems are all interconnected with each other and are seen increasingly. Kubernetes is an up-and-coming container orchestrator which might increase the efficiency of IoT systems.

Security in IoT systems is one of the hot topics of the past decade. Protecting these systems is a challenge as multiple different components need to communicate with each other through mostly wireless communication lines. These communication lines can be intercepted by others leading to theft, damage, and subscription data attack [12].

To test security, a much-used method is penetration testing. Penetration testing is performing an active and authorized simulated cyberattack. The goal of penetration testing is to discover hidden vulnerabilities as quickly as possible. This is mostly done in a safe and protected environment [23].

The automation of penetration testing is a new field in academia. This process is very time-consuming and very expensive due to the need for human intervention. There are a lot of studies in which the automation of penetration testing is discussed, yet this still isn't the conventional way of testing a system's security. There are a lot of uncertainties that need to be accounted for and in this paper, these uncertainties will be discussed [17].

This study aims to find out whether the combination of the automation of penetration testing and IoT systems with a deployed Kubernetes is feasible. This paper will explain the possibilities of Kubernetes with IoT, its future, and whether it can be secured using automatic penetration testing.

A. Research Questions

Can the penetration testing process for the security of an IoT system with K3s deployed be automated?

- What are the current limitations and challenges in automating the penetration testing process and how can they be bypassed?
- What are the challenges of deploying a lightweight version of Kubernetes onto IoT systems, what are its requirements, and is it feasible?

B. Methodology

To answer our research questions, an IoT system needs to be built. This will be done with the use of Raspberry Pi (IoT system) and K3s (lightweight version of Kubernetes). After this system is up and running, a vulnerability scenario needs to be created to start a penetration testing process on the system. This must be done manually at first and all steps need to be clearly denoted. Afterward, a reflection must be done on these steps before we start the automation. All steps must be checked for the automation options, are all of them feasible, and if not, what are their restraining factors? If the (partial) automation is possible, a script will be made to check if the results are correct. Lastly, these results need to be reflected upon for future research purposes.

II. Related Work

The used articles were mainly gathered through Google Scholar, IEEE, and the official websites of the used software. Plenty of studies have appeared around Kubernetes and as it is a new development, most of these were relevant to my research.

In Table I, all used sources are sorted by theme. Some important pieces for my research included the work of S. Böhm et al. [7] who presented the difference between all different versions of Kubernetes. The paper of L. Yan et al. [12] provided a lot of insight into the most relevant parts of IoT and their security. The website of K3s themselves provided very good data about their machines and their hardware usage. Which was essential to answer my research questions. Lastly, J. Liu et al. [23] did a lot of research on the automation of penetration testing.

TABLE I
CLASSIFIED CITATIONS BY THEME

Article	Description
[1], [3], [4], [6], [7], [11], [15], [16], [19]	Kubernetes
[9], [10], [18], [21], [24]	IoT and Hardware
[2], [5], [8], [14]	Implementation of Kubernetes in IoT
[12], [13], [20]	Security in IoT
[17], [22], [23]	Penetration Testing

III. KUBERNETES

A. What is Kubernetes?

Kubernetes is an open-source container orchestration system, originally created by Google. It manages containerized applications across multiple hosts for deploying, monitoring, and scaling containers. As Kubernetes has been donated to the Cloud Native Computing Foundation in 2016, the applications are still being discovered [3]. The platform is known for its scalability and flexibility. It is designed to run on a cluster of machines and can efficiently utilize the available hardware resources. The hardware requirements for running Kubernetes include a minimum of 2 CPU cores, 2 GB of RAM, and 20 GB of disk space [19]. However, these are just the minimum requirements, and the actual hardware requirements will depend on the size and complexity of the workload being managed by Kubernetes. As the workload increases, more CPU cores, RAM, and disk space will be required to ensure optimal performance.

Because of these demanding requirements, K3s has been developed. K3s is a lightweight version of Kubernetes that is specifically designed for resource-constrained environments such as edge devices, IoT devices, and small-scale deployments. K3s requires a minimum of 1 CPU core, 512 MB of RAM, and 200 MB of disk space. This might not seem like a huge difference, but as a lot of IoT components want to be as small as possible, $\frac{1}{4}^{th}$ the size is an enormous change. For this research, we will also be using the lightweight version of Kubernetes, K3s. According to [7], the main benefit of using K3s over K8s (Kubernetes) is the reduced amount of disk usage. The usage of other components stays lower, yet similar.

B. Advantages and disadvantages

Kubernetes has many advantages, but also some disadvantages that need to be considered when implementing this container orchestration system.

1) Advantages:

a) Scalability: Kubernetes allows for easy scaling of applications by automatically managing the deployment and scaling containers based on resource utilization. This ensures that applications can handle increased workloads without downtime or performance degradation [14].

b) Fault-tolerance: Kubernetes provides fault-tolerance by automatically monitoring the health of containers and restarting or replacing them whenever they fail. This ensures that applications remain available and resilient to failures [9].

c) Resource efficiency: Kubernetes optimizes resource utilization by dynamically allocating resources to containers based on their needs. This helps to maximize the utilization of hardware resources and helps reduce costs [14].

d) Portability : Kubernetes provides a platform-agnostic infrastructure for deploying and managing applications. This allows applications to be easily moved between different environments, such as on-premises and cloud, without requiring significant modifications [16].

2) Disadvantages:

a) Complexity: Kubernetes has a steep learning curve and requires a deep understanding of its concepts and components. Setting up and managing a Kubernetes cluster can be complex and time-consuming [16]. The networking itself can also be very complex, especially in multi-cluster or hybrid cloud environments. Configuring and managing network connectivity between containers and services can be challenging [8].

b) Resource-demanding: Kubernetes introduces additional resource overhead due to the need for managing containers and orchestrating their deployment. This can result in increased resource consumption compared to running applications directly on virtual machines or direct hardware [16].

c) Unreliable communication links: In edge computing environments, there may be unreliable communication links between edge nodes. This can impact the load-balancing capabilities of Kubernetes [15].

The advantages are outweighing the disadvantages. However, the resource-demanding aspect of Kubernetes could be a big factor in the consideration of Kubernetes deployment in IoT as those systems try to be as compact as possible.

IV. HARDWARE IN IOT

A. IoT requirements

Before the research question can be answered, we must understand the requirements and goals of an IoT system. An IoT (Internet of Things) system is characterized by the integration of physical devices, sensors, and actuators with the Internet and cloud-based services, enabling them to communicate and exchange data with each other. An IoT system contains four main layers: [11]

- Sensing Layer: Responsible for collecting data by using sensors and actuators.

- Network Layer: Providing information and connectivity between devices in IoT systems.
- Service Layer: Data processing of all received data.
- Interface layer: The interaction between the user and the system.

The network layer is the most important for our research. The network layer in IoT refers to the communication infrastructure that connects the IoT devices and enables data transmission. It plays a crucial role in the performance and efficiency of the overall IoT system and is especially important when trying to implement Kubernetes as Kubernetes is very hardware-demanding. The sensing layer and network layer are the main layers that have compactness requirements, the service layer could in theory be bigger and have fewer limitations.

IoT is a very broad topic and the hardware needs of IoT systems vary depending on the specific application and the technologies used. In general, IoT systems require efficient and fast communication between devices, which is facilitated by messaging protocols. The choice of messaging protocol can significantly impact the performance of IoT systems [10]. Other than that, some IoT systems need to process more information than others, need to be more compact, or have more devices connected. There are a lot of factors to take into account and these requirements are incomparable. One thing that holds value for all systems, is the hardware performance improvement.

B. IoT hardware developments

1) Moore's law: Moore's law is a techno-economic model discovered by Gordon Moore that has stood ground for decades. He found patterns in the improvement rate of the performance in technology and time. According to Moore's law, the performance and functionality of digital electronics doubled roughly every 2 years within a fixed cost, power, and area. Forecasts have been made previously that the law would find its demise over time, yet every time they have consistently been wrong [18]. Again, there are people trying to prove that Moore's law might be coming to an end and that the performance increase might start to plateau. Aside from some current limitations, no proof is provided yet.

2) IoT hardware developments: Looking toward the future, the development of IoT systems is closely tied to advancements in communication technologies. The upcoming 5G network technology is expected to bring significant improvements in data transfer speed and bandwidth, which will have a positive impact on IoT applications [21]. Additionally, the integration of technologies like artificial intelligence, big data, and blockchain will further enhance the capabilities of IoT systems [20]. Other than 5G network technology, cryptographic accelerators can greatly improve the efficiency of cryptographic functions [13]. These technologies are already around but have yet to be implemented in IoT.

V. Kubernetes and IoT

As stated earlier, K3s' minimum requirements are 1 CPU core, 512 MB of RAM, and 200 MB of disk space. For this research, K3s was deployed on a Raspberry Pi 4 Model B [1]. The Raspberry Pi could handle this without any issues. According to the website of K3s themselves, the container orchestrator uses the resources shown in Table II. These tests were performed on a Raspberry Pi 4 Model B with the components shown in Table III. [4]

TABLE II
Resource Usage K3s on Raspberry Pi 4 Model B

Components	Min CPU	Min RAM
K3s server with a workload	30% of a core	1588 M
K3s cluster with a single agent	25% of a core	1215 M
K3s agent	10% of a core	268 M

TABLE III
Raspberry Pi 4 Model B details

Arch	OS	CPU	RAM	Disk
aarch64	Raspberry Pi OS 11	BCM2711, 4 Core 1.50 GHz	8 GB	USH-III SDXC

These results show that the requirements of running K3s are very minimal and that there is plenty of space to include security measures. This was one of the main concerns as an important part of Kubernetes, as well as IoT, is the transmission of information between different containers or devices. These communication lines will have to be encrypted in some way if they want to be secure. There are many options out there that provide these security measures that do not take up a lot of hardware usage. K3s has some security mitigations applied by default yet allows the implementation of other options as well. [2] [5]

The K3s agents are using much fewer resources than the servers. These server nodes regulate everything and do not necessarily have to be as compact as the agents. One of the IoT systems' requirements could be the compactness of certain components. The fact that K3s agents do not necessarily need a lot of computational power, provides this option. One of the additional requirements may be that the K3s server component has to be a little bigger.

VI. Automated Penetration Testing

A. Security in IoT

In today's digital age, where technology permeates almost every aspect of our lives, the importance of security cannot be overstated. IoT devices collect vast amounts of personal data, often without users' explicit knowledge or consent. This data should be secured properly as it could otherwise be misused, leading to serious privacy breaches. According to a study conducted by the International Data Corporation (IDC), by 2025, the number of connected IoT devices is expected to reach 150 billion globally [24]. Such a massive influx of

devices necessitates proactive measures to protect user privacy and prevent unauthorized access to sensitive information.

As IoT systems have an interconnected nature as they consist of separate communicating components, they are an attractive target for hackers. These different connections are weaknesses and are tough to protect well. Unsecured IoT devices can serve as entry points for cyberattacks, leading to severe consequences such as data breaches, network intrusions, and even physical harm.

B. Penetration Testing

Traditionally, systems tended to have reactive security measures. They tried to build flawless systems and fixed flaws found by hackers. This was very inefficient and could be very costly. Over time, penetration testing started to become more and more popular. Penetration testing is a proactive measure to protect a system by attacking it. Essentially, an attack will be performed on a system to try to cause harm. If there are To have a highly rated secure system, near-flawless penetration testing is needed. Every fault is an entrance for intruders. [22]

As stated, penetration testing consists of active attacks. This means that this cannot be set up passively and that it is hard to automate. Every penetration testing attempt has to be set up manually due to the complexity and diversity of IoT systems. IoT systems already consist of a wide range of devices, protocols, and communication channels. This makes it difficult to develop a comprehensive and standardized automated penetration testing approach. Additionally, the heterogeneity of IoT systems can result in compatibility issues with automated penetration testing tools, limiting their effectiveness.

C. Attacking K3s deployed IoT system

With the use of the work of M. Akula [6], vulnerable scenarios could be deployed on the K3s deployed IoT system. This git-hub user provides a 'playground' which showcases the common misconfigurations, real-world vulnerabilities, and security issues in Kubernetes clusters, containers, and cloud-native environments.

The given scenarios on Kubernetes Goat, M. Akula's 'playground', provides a system with some specific weaknesses. The goal is to retrieve 'secret' information by attacking the system on one of its weak spots. Individually, all these scenarios could be solved with relative ease. The automation of penetration testing to check if one part of the system is vulnerable inquires that all these attacks should be performed automatically. After which it can show which attacks work and which do not.

To manually perform these attacks, a few simple lines of code will do the trick. To start, multiple solutions to different scenarios were added after one another in a Python script. When ran, this script slowly tried multiple different possible attacks and retrieved the results of all of the included attacks. This approach worked but was fairly inefficient and retrieved a lot of unnecessary data. To start to increase efficiency and provide a clearer overview, different required components were installed at the beginning of the script. Some of these components were used in several of the attempted attacks. This improvement decreased the run time of the script a little but is assumed to have a bigger effect on a larger scale. These components that needed installation could eventually be incorporated into the final script if wished.

Even though we are looking for weaknesses only, this script presents everything it found at the end. This includes a lot of information that is not relevant to the attacker. The user of the script did not have to specify what is being looked for initially. To fix this, all results from all attacks were linked to each other. Whenever the attacker found a result that was looked for, the attack and its corresponding weak part of the system were linked there.

This was much better, but could still use some improvements. Eventually, a new option was added where the attacker can search for some value, and the script would run and return the attack which resulted in this value if found. This only works if you specifically know the value you're looking for. But if the attacker does not know which value is being looked for, it gets much more difficult. Normally this would require human interaction, someone who notices what seems to be off and insecure. This step has not been automated in this research but is very important for its automation.

All these attacks work automatically on this specific setup with the Raspberry Pi 4 Model B. That does not mean that this works on all IoT systems that have K3s deployed. A lot of variables have to be altered in the scripts and some attacks required an attentive eye.

It is important to note that some attacks were not performed. For example, the DoS (Denial of Service) attack on the memory/CPU resources as this attack would stop the Raspberry Pi from functioning which does not combine well with the other attacks. As only a small part of the possible security risks was checked, the duration of the script is not relevant to our results.

VII. Discussion

It has become clear that K3s is a feasible option for IoT systems. Especially when realizing that these technologies are still improving quite rapidly. The encryption of these systems is possible as of now but might require improvements as attacks get better and better. As for now, these should be trouble-free when used by an average IoT user. However, bigger companies, or people concerned more about their security, might want to wait till more information is known about these systems and their security.

Penetration testing is easily done manually on these IoT systems with K3s deployed. And even though the automation of this is possible, a lot of improvements can be made. These improvements will not be easy as these scripts will have to find out which information is useful or wanted. If this cannot be done automatically by an algorithm, a lot of time will be wasted going through all the results manually. Even though

some conveniences were added to the basic script, this does define a fully automated penetration testing attack.

VIII. Conclusion and future research

K3s is a very good option to use in IoT systems. There might still be some issues, but overall, it already works on the existing hardware. The main problems it could face, concern the hardware capabilities of separate sensors that have a tight required compactness. If the processing of data can be done elsewhere and its performance is capable of running a K3s agent, no issues should be faced. Especially with the still-improving hardware performance according to Moore's Law.

Automation in penetration testing is hard to accomplish as systems differ a lot from each other. A checklist of attacks on different faulty scenarios can be conducted automatically, but this is not flawless. Kubernetes is very complex, and therefore its security is hard to guarantee, and its weaknesses are hard to find with a simple script.

The approach that we took works for this specific IoT system, but this will vary with different systems. There are a lot of variables that need to be accounted for but have not been automated in this script. Additionally in the current automation, we pass all different attacks one by one. This could be optimized by combining different attacks in similar areas.

For future research, the compactness and automation of penetration testing could be discovered further. In our case, the efficiency of the script did not matter. But this could be interesting when done on a larger scale. Which parts take up a lot of time and how can they be improved? Lastly, a lot of attacks require that the attacker knows what he is looking for. The current solution cannot do that well, perhaps an algorithm could find these relations and attack more specifically. These different approaches could potentially solidify this method of securing IoT systems if done properly.

IX. Acknowledgment

I would like to express my deepest appreciation to my supervisor S. Simonetto and my coordinator dr. ir. A. Chiumento for providing me with the opportunity to conduct this research and for their guidance throughout.

References

[1] The certified kubernetes distribution built for iot edge computing. URL: https://k3s.io/.

[2] Hardening guide. URL: https://docs.k3s.io/reference/resource-profiling.

[3] Production-grade container orchestration. URL: kubernetes.io.

[4] Resource profiling. URL: https://docs.k3s.io/reference/resource-profiling.

[5] Secrets encryption config. URL: https://docs.k3s.io/reference/resource-profiling.

[6] M. Akula. Interactive kubernetes security learning playground. URL: https://madhuakula.com/kubernetes-goat/.

[7] S. Böhm and G. Wirtz. Profiling lightweight container platforms: Microk8s and k3s in comparison to kubernetes. In *ZEUS*, pages 65–73, 2021.

[8] Costa-Requena-J. Ivanciu I. Strautiu-V. Dobrota V. Botez, R. Sdn-based network slicing mechanism for a scalable 4g/5g core network: A kubernetes approach. *Sensors*, 21:3773, 2021. https://doi.org/10.3390/s21113773 doi:10.3390/s21113773.

[9] Razik-L.-Monti A. Dähling, S. Enabling scalable and fault-tolerant multi-agent systems by utilizing cloud-native computing. *Auton Agent Multi-Agent Syst*, 35, 2021. https://doi.org/10.1007/s10458-020-09489-0 doi:10.1007/s10458-020-09489-0.

[10] Fernández-Caramés T. M. Fraga-Lamas P. Castedo L. Froiz-Míguez, I. Design, implementation and practical evaluation of an iot home automation system for fog computing applications based on mqtt and zigbee-wifi sensor nodes. *Sensors*, 18:2660, 2018. https://doi.org/10.3390/s18082660 doi:10.3390/s18082660.

[11] Bhat O.-Bhat S. Gokhale, P. Introduction to iot. *International Advanced Research Journal in Science, Engineering and Technology*, 5(1):41–44, 2018.

[12] Yan L.-Liu Y. Li Y. Gou, Q. Construction and strategies in iot security system. In *2013 IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, pages 1129–1132, 2013. https://doi.org/10.1109/GreenCom-iThings-CPSCom.2013.195 doi:10.1109/GreenCom-iThings-CPSCom.2013.195.

[13] Lettieri G.-Perazzo P. Saponara S. Leonardi, L. On the hardware–software integration in cryptographic accelerators for industrial iot. *Applied Sciences*, 12:9948, 2022. https://doi.org/10.3390/app12199948 doi:10.3390/app12199948.

[14] Abbas S.-Soliman A. Alyas T.-Asif S. Faiz T. Niazi, M. Vertical pod autoscaling in kubernetes for elastic container collaborativeramework. *Computers Materials Continua*, 74:591–606, 2023. https://doi.org/10.32604/cmc.2023.032474 doi:10.32604/cmc.2023.032474.

[15] Kundroo M.-Park D. Kim S.-Kim T. Phuc, L. Node-based horizontal pod autoscaler in kubeedge-based edge computing infrastructure. *IEEE Access*, 10:134417–134426, 2022. https://doi.org/10.1109/access.2022.3232131 doi:10.1109/access.2022.3232131.

[16] Kienast P.-Vinogradov G. Ganser P.-Bergs T. Rudel, V. Cloud-based process design in a digital twin framework with integrated and coupled technology models for blisk milling. *Front. Manuf. Technol.*, 2, 2022. https://doi.org/10.3389/fmtec.2022.1021029 doi:10.3389/fmtec.2022.1021029.

[17] Buffet O.-Hoffmann J. Sarraute, C. Pomdps make better hackers: Accounting for uncertainty in penetration testing. *CoRR*, abs/1307.8182, 2013. URL: http://arxiv.org/abs/1307.8182, http://arxiv.org/abs/1307.8182 arXiv:1307.8182.

[18] R.R. Schaller. Moore's law: past, present and future. *IEEE Spectrum*, 34(6):52–59, 1997. https://doi.org/10.1109/6.591665 doi:10.1109/6.591665.

[19] Li Q.-Kraft P. Kaffes K.-Hong D. C. Mathew S. ... Zaharia M. Skiadopoulos, A. Dbos. *Proc. VLDB Endow.*, 15:21–30, 2021. https://doi.org/10.14778/3485450.3485454 doi:10.14778/3485450.3485454.

[20] J. Soldatos. Security risk management for the internet of things: Technologies and techniques for iot security, privacy and data protection. 2020. https://doi.org/10.1561/9781680836837 doi:10.1561/9781680836837.

[21] Shathik J.-Prasad K. Vikranth, K. Future enhancements and propensities in forthcoming communication system – 5g network technology. *J. Phys.: Conf. Ser.*, 1712:012006, 2020. https://doi.org/10.1088/1742-6596/1712/1/012006 doi:10.1088/1742-6596/1712/1/012006.

[22] C. Weissman. Penetration testing. *Information security: An integrated collection of essays*, 6:269–296, 1995.

[23] Liu J.-Hou D. Zhong X.-Zhang Y. Zhou, S. Autonomous penetration testing based on improved deep q-network. *Applied Sciences*, 11(19):8823, 2021. URL: https://www.proquest.com/scholarly-journals/autonomous-penetration-testing-based-on-improved/docview/2580965017/se-2.

[24] Weatherill L. Zwolenski, M. The digital universe: Rich data and the increasing value of the internet of things. *Journal of Telecommunications and the Digital Economy*, 2:9, 05 2020. https://doi.org/10.18080/jtde.v2n3.285 doi:10.18080/jtde.v2n3.285.