RIoTPot: a modular hybrid-interaction IoT/OT honeypot

Shreyas Srinivasa, Jens Myrup Pedersen, and Emmanouil Vasilomanolakis

Aalborg University, Denmark
{shsr,jens,emv}@es.aau.dk

Abstract. Honeypots are often used as a proactive attack detection mechanism and as a source of threat intelligence data. However, many honeypots are poorly maintained and cumbersome to extend. Moreover, low-interaction honeypots are prone to fingerprinting attacks due to their limited emulation capabilities. Nonetheless, low-interaction honeypots are essential for environments with limited resources. In this paper, we introduce RIoTPot, a modular and hybrid-interaction honeypot for Internet-of-Things (IoT) and Operational Technology (OT) protocols mainly used in Industrial Control System environments. RIoTPot's modularity comes as a result of plug-n-play container services while its hybrid-interaction capability enables users to switch between low- and high-interaction modes. We deploy RIoTPot on the Internet, receive a large amount of attacks and discuss the results received on both low- and high-interaction modes.

1 Introduction

Honeypots are deceptive systems that simulate a seemingly vulnerable system to gather attacks. Over the years, many honeypot solutions have been proposed that are commonly classified to low-, medium- and high-interaction based on the level of interaction they offer to the adversary [19, 2]. Low- and medium-interaction honeypots, due to their limited emulation capabilities, are prone to honeypot fingerprinting that may limit their scope [18]. Honeypot fingerprinting refers to adversarial methods that allow for the identification of the honeypot nature of a system. Nevertheless, these two classes of honeypots are the most commonly deployed ones. Other common issues with honeypots include the lack of flexibility in extending/adapting them, the absence of support, and limited documentation.

Despite the aforementioned limitations, honeypots are an excellent defensive toolkit, especially with regard to the increasing number of IoT and OT attacks. With such protocols being consistently attacked, in both consumer [7] and commercial environments [5], deception mechanisms like honeypots offer an early warning system and a method to analyse adversaries' techniques [16, 12, 9].

Traditional honeypot simulations may run on virtualized environments like VMs, virtual containers (LXC), or even language-based virtual environments.

Kedrowitsch et al. made a first effort to propose the use of containers for honeypots [6]. The authors propose the usage of Linux containers as a platform to develop honeypots and compliment their proposal by comparing the detection methods of popular virtualization platforms against containers. Kedrowitsch et al. conclude that limitations exist in the use of either containers or virtual machines as a honeypot platform. A much recent proposal by Reti et al. introduces the use of container-based deception for honeypots [11]. The authors investigate the possibilities of container-based honeypots and introduce the concept of simulating container-escapes (fake network pivoting outside a container) as a deception technique. Both approaches suggest the use of container systems to achieve ease of deployment. Moreover, many open-source honeypots offer the possibility of a containerized deployment for ease of installation. Nevertheless, besides the aforesaid academic work there are not many actual honeypot implementations that make use of containers. Furthermore, all existing honeypots have a binary interaction level: they are either low-, medium-, or high-interaction [19].

In this paper, we present RIoTPot[1], a honeypot that: *i.)* breaks the traditional binary interaction paradigm, *ii.)* focuses on IoT and OT protocols, and *iii.)* is designed with a modular-by-design architecture. First, the hybrid-interaction level of RIoTPot aims at providing defenders flexibility by giving them the ability to utilize the appropriate interaction level based on their needs and capabilities. For instance, low constrained environments scale better with low interaction components while high interaction comes handy when deeper analysis of attack is required. Second, RIoTPot supports many IoT and OT protocols (i.e., Telnet, SSH, CoAP, Modbus, MQTT), with more to be implemented in the immediate future. At the moment, there are only a few real world honeypot implementations that focus on IoT [15, 9] and even fewer for OT [12, 17]. Lastly, the modularity of the honeypot comes from its architecture; each functionality of the honeypot is a plug-n-play component that can be edited, activated or deactivated based on the user's preferences.

2 RIoTPot Design

RIoTPot features a modular architecture that facilitates quick integration of new protocol simulation modules. A modular software architecture is a structural approach of building software components as modules by separating the the functionality of a program into independent, interchangeable modules, such that each contains everything necessary to execute only one aspect of the desired functionality [8]. Figure 1 shows the high level architecture of RIoTPot. The prominent modules in the architecture are the *RIoTPot core* module, the *packet capture and noise filter* module, the *low-interaction* modules, the *high-interaction* modules, and the *attack database*.

The *RIoTPot core* consists of the required components for the configuration, administration, and orchestration of the honeypot. In particular, the core module

[1] https://github.com/aau-network-security/riotpot

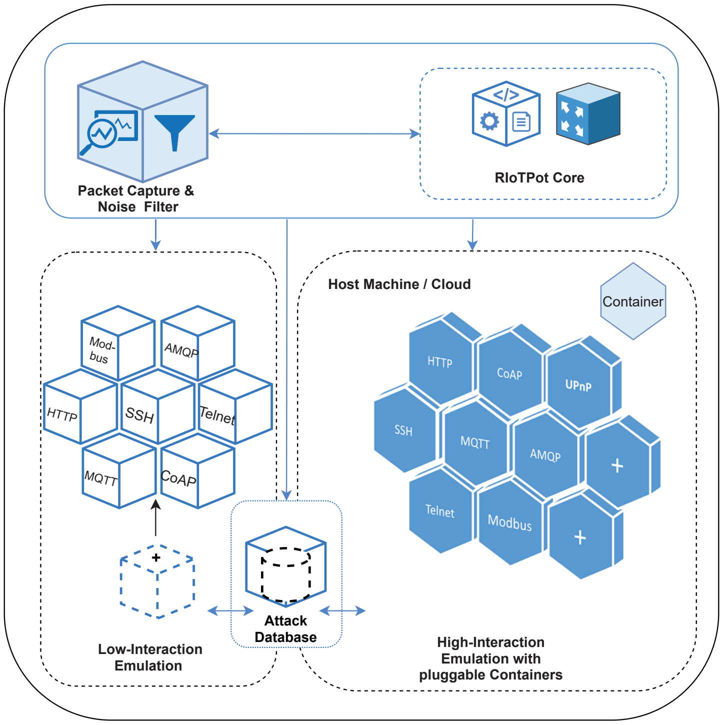

Fig. 1. High level architecture of RIoTPot

provides RIoTPot with all the required parameters at startup. This includes user preferences for specific protocols, profile simulation, and the desired interaction level. In addition, the core is responsible for the network management for the high-interaction protocol services simulated on containers. The received attack traffic is forwarded to the respective container that hosts the protocol on which the attack was targeted. Furthermore, the core also facilitates the communication between itself and the containers, if hosted on a cloud environment.

For the *Packet capture and noise filter* module the attack capture component is responsible for storing the attack packets as *pcap* files, using *tcpdump*, which can be used for detailed analysis (e.g., deep packet inspection). The noise filter component filters out the traffic received from Internet-wide scanners like Shodan [14] and Censys [3]. This helps the honeypot administrator to concentrate on attacks that matter by removing the noise traffic generated by such services.

The *low-interaction mode* is achieved through independent packages, with each package simulating a specific protocol. RIoTPot is implemented in Go language [4] and facilitates the development of a modular architecture through packages. The packages act as plug-ins that can be added to the honeypot to extend the protocols simulated. For example, the *fakeshell* package emulates a system shell that can be leveraged by the SSH and the Telnet packages. The *fakeshell* package can be extended to include emulation of specific commands. Furthermore, RIoTPot provides a template that can be used for integration of additional protocols. The *high-interaction mode* is achieved by emulating the protocols as services in container images. Hence, since a container implements the full protocol the honeypot provides the attacker with high interaction capabilities. The containers act as high-interaction modules that offer a full implementation of a protocol. Additional protocol services can be added by integrating containers with the desired protocol services. The hybrid-interaction mode further allows the user to emulate selective protocols on low or high-interaction levels. For example, the user can choose to have SSH in low-interaction mode and MQTT in high-interaction mode.

The *attack database* stores all the attack traffic received on the honeypot. The database is setup as an independent module to ensure data availability even if a honeypot module is down (e.g., due to a crash or DDoS attack). The database is accessible from the low-interaction and high-interaction modules for attack storage.

To sum up, the design of RIoTPot facilitates modularity through packages and containers as plugins. Furthermore, the modular architecture assists the hybrid-interaction model of RIoTPot.

3 Preliminary Results

The honeypot was deployed in both low and high interaction modes on two hosts in our lab. The hosts were assigned a public IP each, under an unfiltered network. We define an attack as any interaction with the honeypot as there is no production value whatsoever. However, we differentiate incoming traffic from well-known crawlers (e.g. Shodan). The attacks on the honeypots were recorded for a period of one week. In the low-interaction variant, the protocols SSH, Telnet, HTTP, MQTT, CoAP and Modbus were simulated through the plug-in packages, while the high-interaction variant simulated the MQTT protocol in a container. In addition to recording the attacks in the database, the hosts also had the *tcpdump* service running in the background to capture the attack packets for comprehensive analysis. A total of $7,587$ attacks were observed across all the protocols simulated by RIoTPot.

Figure 2 shows the number of unique attacks received per protocol for a period of one week. MQTT-HI indicates the high-interaction mode of the MQTT protocol. We observe a trend in the number of attacks for all protocols. Furthermore, the number of attacks on the MQTT protocol in the high-interaction mode is higher in comparison to the low-interaction mode. Moreover, we observe recurring sessions from same suspicious actors on the high-interaction mode, that included topic creation, subscription and deletion, and modification of existing messages in topics which have not been observed on the low-interaction mode.

Figure 3 depicts the percentage of attacks from Internet-scanning engines (e.g., Shodan, Censys, Project Sonar [10], and ShadowServer [13]) in comparison to the attacks from suspicious sources. We observe an average of 25% of the total

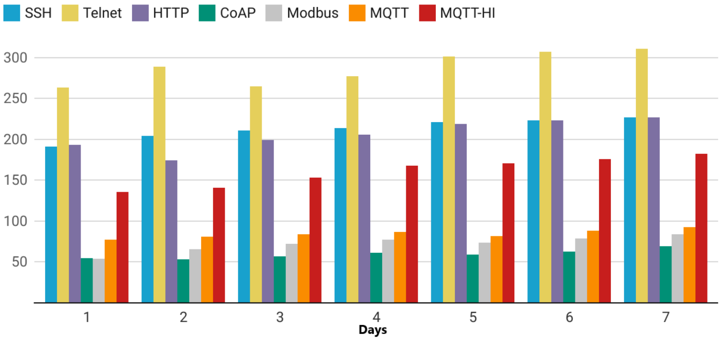

Fig. 2. Number of attacks on protocols per day

traffic originating from 19 common scanning engines[2]. Filtering out such traffic reduces noise and alert data fatigue for the administrators.

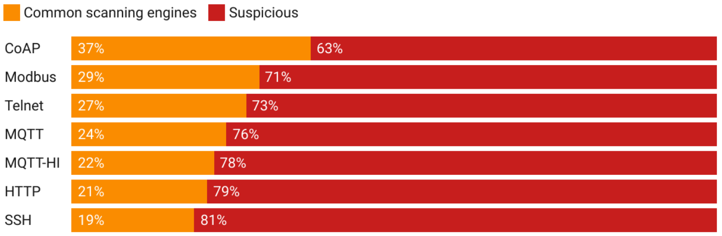

Fig. 3. Attack noise classification in percentage

4 Conclusion

In this paper, we introduce RIoTPot, a honeypot that features a hybrid-interaction model with a modular design for IoT and OT protocols. RIoTPot addresses the issue of limited interaction and flexibility, in addition to ease of deployment. Our preliminary results suggest that the honeypot is attractive to adversaries and is able to distinguish between suspicious traffic (traffic originating from attackers) and common scanning engines (traffic likely coming from Shodan-like systems). As future work, we aim to extend RIoTPot to support more IoT and OT protocols like UPnP, AMQP, XMPP, S7, DNP3, Fieldbus and Profibus. Furthermore, we intend to integrate threat intelligence reporting through STIX to facilitate structured sharing of threat data [1]. Finally, we plan to perform a more extensive evaluation of RIoTPot with an emphasis on ICS environments.

[2] For a complete list of the supported scanning engines see: https://github.com/aau-network-security/riotpot#12-Noise-Filter

References

1. Barnum, S.: Standardizing cyber threat intelligence information with the structured threat information expression (stix). Mitre Corporation **11**, 1–22 (2012)
2. Bringer, M.L., Chelmecki, C.A., Fujinoki, H.: A survey: Recent advances and future trends in honeypot research. International Journal of Computer Network and Information Security **4**(10), 63 (2012)
3. Censys: Censys search (2021), `https://censys.io/`
4. Golang: Go language (2021), `https://golang.org/`
5. Jiang, X., Lora, M., Chattopadhyay, S.: An experimental analysis of security vulnerabilities in industrial iot devices. ACM Trans. Internet Technol. **20**(2) (May 2020). https://doi.org/10.1145/3379542, `https://doi.org/10.1145/3379542`
6. Kedrowitsch, A., Yao, D.D., Wang, G., Cameron, K.: A first look: Using linux containers for deceptive honeypots. In: Proceedings of the 2017 Workshop on Automated Decision Making for Active Cyber Defense. p. 15–22. SafeConfig '17, Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3140368.3140371, `https://doi.org/10.1145/3140368.3140371`
7. Mangino, A., Pour, M.S., Bou-Harb, E.: Internet-scale insecurity of consumer internet of things: An empirical measurements perspective. ACM Transactions on Management Information Systems **11**(4) (Oct 2020). https://doi.org/10.1145/3394504, `https://doi.org/10.1145/3394504`
8. Mohammed, M., Elish, M., Qusef, A.: Empirical insight into the context of design patterns: Modularity analysis. In: 2016 7th International Conference on Computer Science and Information Technology (CSIT). pp. 1–6 (2016). https://doi.org/10.1109/CSIT.2016.7549474
9. Oosterhof, M.: Cowrie ssh/telnet honeypot (2016), `:https://github.com/micheloosterhof/cowrie`
10. Research, R.: Project sonar (2021), `https://www.rapid7.com/research/project-sonar/`
11. Reti, D., Becker, N.: Escape the fake: Introducing simulated container-escapes for honeypots (2021)
12. Rist, L., Vestergaard, J., Haslinger, D., Pasquale, A., Smith, J.: Conpot ics/scada honeypot. Honeynet Project (conpot. org) (2013)
13. ShadowServer.org: Shadowserver.org (2021), `https://www.shadowserver.org/`
14. SHODAN: Shodan (2021), `https://www.shodan.io/`
15. Vasilomanolakis, E., Karuppayah, S., Fischer, M., Mühlhäuser, M., Plasoianu, M., Pandikow, L., Pfeiffer, W.: This network is infected: Hostage-a low-interaction honeypot for mobile devices. In: Proceedings of the Third ACM workshop on Security and privacy in smartphones & mobile devices. pp. 43–48 (2013)
16. Vasilomanolakis, E., Karuppayah, S., Mühlhäuser, M., Fischer, M.: Hostage: A mobile honeypot for collaborative defense. In: Proceedings of the 7th International Conference on Security of Information and Networks. p. 330–333. SIN '14, Association for Computing Machinery, New York, NY, USA (2014). https://doi.org/10.1145/2659651.2659663, `https://doi.org/10.1145/2659651.2659663`
17. Vasilomanolakis, E., Srinivasa, S., Mühlhäuser, M.: Did you really hack a nuclear power plant? an industrial control mobile honeypot. In: 2015 IEEE Conference on Communications and Network Security (CNS). pp. 729–730. IEEE (2015)

18. Vetterl, A., Clayton, R.: Bitter harvest: Systematically fingerprinting low- and medium-interaction honeypots at internet scale. In: 12th USENIX Workshop on Offensive Technologies (WOOT 18). USENIX Association, Baltimore, MD (Aug 2018), `https://www.usenix.org/conference/woot18/presentation/vetterl`
19. Zhang, L., Thing, V.: Three decades of deception techniques in active cyber defense - retrospect and outlook. Computers & Security **106**, 102288 (2021). https://doi.org/https://doi.org/10.1016/j.cose.2021.102288, `https://www.sciencedirect.com/science/article/pii/S0167404821001127`