Decepti-SCADA: A Framework for Actively Defending Networked Critical Infrastructures

Nicholas Cifranic[1], Jose Romero-Mariona[1], Brian Souza[1] and Roger A. Hallman[1,2]
[1]*Naval Information Warfare Center Pacific, San Diego, California, U.S.A.*
[2]*Thayer School of Engineering, Dartmouth College, Hanover, New Hampshire, U.S.A.*

Keywords: Cybersecurity, Deception, Honeypots, Cyber-physical Systems, Supervisory Control and Data Acquisition (SCADA), Industrial Control Systems (ICS), Critical Infrastructure.

Abstract: Supervisory Control and Data Acquisition (SCADA) networks, which enable virtual components of critical infrastructures to connect to physical components, like the electrical grid, for example, are susceptible to cyber threats. This introductory paper discusses the application of deception as a technique for improving the cybersecurity posture of a network by using decoys to obfuscate the network and in turn make it harder for a potential adversary to find the real components. The Decepti-SCADA framework is introduced, which demonstrates multiple improvements over previous implementations of cyber deception strategies for SCADA systems. Decepti-SCADA has developed SCADA-specific decoys that can be used in a critical infrastructure environment. We detail Decepti-SCADA's architecture, decoy generation and distribution, and ultimately explore what else can be done with cyber deception for critical infrastructures through early results.

1 INTRODUCTION

The current state of computer security is an asymmetric battle between defenders and attackers. Defenders need to have a high level of security assurance on their networks and that requires an intensive risk management process. When securing a computer network, it is impossible to eliminate all vulnerabilities so defenders have to focus on minimizing their risk (Denning and Denning, 2016). The way to minimize risk is to find what vulnerabilities are present on their systems, then rate the likelihood and the impact of those vulnerabilities being exploited. That leaves defenders with a model of which vulnerabilities are a priority and need to be fixed first. Defenders have limited recourses so they must choose what fixes to implement based on the cost price analysis of said fixes. This leaves some lower priority vulnerabilities unpatched, and other vulnerabilities with imperfect fixes. In short, completely defending an operational network is an impossible task. In order for attackers to accomplish their goals, they don't need to find all the weak points in a network, they only need to find a few. In this way, attackers have an asymmetric advantage.

Defenders add security features to their networks like firewalls and intrusion detection systems in order to prevent unauthorized and unwanted access to their systems. These types of security features can successfully thwart cyber-attacks, but are fundamentally honest. When attackers do network reconnaissance, they have a high level of confidence that their interactions with the systems on the network can be trusted as legitimate. Defenders can take advantage of this trust by adding deception to their network. Deception can be used to degrade the quality of the attacker's understanding of the network that they are trying to exploit. This will cause an attacker to spend a disproportionate amount of time interacting with devices that don't add useful information to the attacker. Any amount of time that an attacker is interacting with a decoy is less time with a real machine and less progress towards their goals.

In this paper we present the Decepti-SCADA framework, which utilizes recent advances in deception and cybersecurity and integrates them into the realm of operational and industrial technologies. Our main contribution is the introduction of a high-interaction honeypot system for networked critical infrastructure. Furthermore, Decepti-SCADA has a modular, Dockerized design (Merkel, 2014) that makes it much more user-friendly than currently existing implementations of deception technologies for industrial control systems.

Cifranic, N., Romero-Mariona, J., Souza, B. and Hallman, R.
Decepti-SCADA: A Framework for Actively Defending Networked Critical Infrastructures.
DOI: 10.5220/0009343300690077
In *Proceedings of the 5th International Conference on Internet of Things, Big Data and Security (IoTBDS 2020)*, pages 69-77
ISBN: 978-989-758-426-8

The remainder of this paper is organized as follows: Section 2 provides background information on deception and SCADA systems which will be necessary for developing an understanding of the Decepti-SCADA framework. Sections 3, 4, and 5, respectively, present the Decepti-SCADA framework, architecture, and deployment. Concluding remarks and future work are discussed in Section 6.

2 BACKGROUND

We begin by covering background information that lays the foundation for the Decepti-SCADA framework. Specifically, we provide a brief history of deception and survey its application to cybersecurity for information technology (IT) systems. We also discuss SCADA networks and how these operational technology systems differ from more well-known IT systems. Readers wishing for more detailed surveys of the topics are encouraged to read (Pawlick et al., 2019) for deception in cybersecurity and (Arghira et al., 2011; Nazir et al., 2017) for an overview of modern SCADA systems and their specific cybersecurity needs, respectively.

2.1 Deception in Cybersecurity

Communication, be it between people or machines, operates on the presumption of truthfulness. Deception occurs when the message sender intentionally transmits a false message to the receiver, with the intent of fostering an inaccurate belief state (Buller and Burgoon, 1996). Indeed, deception has been used in adversarial situations throughout history to create a strategic advantage where it may not have previously existed (Latimer, 2003). Examples of deception tactics include the creation of a fake Allied invasion force in Kent during World War II (Wheatley, 1976) or 'bluffing' in games of poker (Palomäki et al., 2016). In the case of World War II's fake invasion force, the Allies' goal was to fool NAZI forces into relocating their forces and making the actual landing site on Omaha Beach a softer target; poker players typically bluff when they they have a weak hand and want to convince their opponent to fold.

Where traditional (i.e., passive) cybersecurity is primarily oriented towards intrusion prevention and detection, deception is categorized as a form of active security (Denning, 2014). Deception for cybersecurity seeks to manipulate the information that an adversary learns during the attack reconnaissance phase (Bilinski et al., 2019). Pawlick, et al (Pawlick et al., 2019), provide a detailed taxonomy of defensive deception:

- *Perturbation* is a deception technique which uses the insertion of noise to limit the leakage of sensitive information. Differential Privacy is a notable example of a perturbation strategy (Dwork, 2011).
- *Moving Target Defenses* (MTD) actively reconfigure network assets and defensive tools to impair an adversary's attack reconnaissance (Jajodia et al., 2012).
- *Obfuscation* defenses waste an adversary's resources by presenting and directing them to decoy targets as opposed to the network's actual assets, as well as presenting fraudulent information intermixed with legitimate (i.e., valuable) information (Chakraborty et al., 2019).
- *Mixing* strategies use exchange systems to prevent direct linkage between systems (e.g., the use of the Tor network (AlSabah and Goldberg, 2016)).
- *Honey-x* deception strategies refer to the use of technologies such as honeypots (Fan et al., 2017), honey-patches (Araujo et al., 2014), etc.,that masquerade as legitimate network assets but include advanced monitoring capabilities which enable system administrators to discern information on attackers.
- *Attacker Engagement* involves the use of feedback to influence attacker behavior for extended periods, wasting their resources and allowing network administrators to conduct counterintelligence operations (Gutzwiller et al., 2018; Bilinski et al., 2019).

2.2 Modern SCADA Systems

Supervisory Control and Data Acquisition (SCADA) networks, systems which monitor and control automated processes, are important to both ICS and critical infrastructure. In power generation and distribution infrastructure, SCADA systems are used for supervision, control, optimization, and management of generation and transmission systems (Arghira et al., 2011). SCADA systems typically consist of components such as:

- Remote Terminal Units (RTUs) automatically collect data from collected sensors, meters, or other process equipment;
- Programmable Logic Controllers (PLCs) are used in numerous control applications as well as read meters and application status reports.

Furthermore, SCADA systems have typically used numerous, often proprietary, protocols that limit interconnectivity with other systems (as opposed to more recent trends in protocol standardization).

2.2.1 Cybersecurity for SCADA Systems

SCADA systems are notoriously fragile systems with many legacy devices which are unable to support standard security technologies, and this has led to numerous cybersecurity difficulties (e.g., SCADA systems are susceptible to timing errors due to the interaction with cybersecurity components). Members of the Decepti-SCADA team have previously surveyed the current state of SCADA security and developed evaluation capabilities to grade the suitability of cybersecurity products for SCADA networks (Romero-Mariona. et al., 2016). This review of cybersecurity for SCADA concentrates on defensive deception—honey-x strategies, in particular—applied to ICS and critical infrastructure.

The use of a cyber deception strategy on SCADA systems allows defenders to disrupt an attacker's asymmetric advantage by presenting false information that the attacker accepts as the truth, and build a false belief state based on that information. Thus a deception strategy offers advantages that other defense strategies cannot. Using decoys on a SCADA system increases the attack surface that an attacker must interact with in order to complete their goals. For instance, if an Attacker is doing passive reconnaissance (e.g.,simply monitoring traffic) decoys will obfuscate which values and protocols are actually being used on the real systems, delaying or deterring an attacker. If an attacker is doing active reconnaissance, then deception increases the size of their network footprint by forcing them to have to interact with real devices as well as decoys. Network intrusion detection systems then have a higher probability of detecting attacker, since their attack footprint is larger. Using honeypots accomplishes this with relatively low overhead cost to the defenders, and the honeypots do not interact with the real SCADA devices. This means that properly configured deception can be used on a SCADA system without interrupting normal system functionality.

Honeypots have been implemented in SCADA cybersecurity strategies in both commercial and academic settings. Commercially, HoneyD[1] and Conpot[2] are two well-known, opensource projects. Honeyd is a small daemon that creates virtual hosts on a network which can be configured to run arbitrary services, and their personality can be adapted so that they appear to be running certain operating systems. HoneyD shows how hosts might appear on the network, however these virtual hosts are not very interactive. Conpot is a low interaction server-side ICS honeypot designed to be easy to deploy, modify and extend. Conpot code is difficult to execute due in part to a poor user interface.

The SCADA HoneyNet Project was an early attempt at integrating honeypots into SCADA systems that was able to simulate stack, protocol, and application levels of architecture, as well as some serial ports. The project is no longer maintained, however project artifacts are available in an online repository[3]. The Digital Bond SCADA HoneyNet is another SCADA honeypot effort which has been abandoned and archived online[4]. Digital Bond's approach could simulate a Programmable Logic Controler (PLC) with Modbus/TCP, FTP, Telnet, HTTP, and SNMP available to the attacker. However, a separate machine is required for network monitoring.

More recently, Simoes, et al. (Simões et al., 2013; Simões et al., 2015), consider the use of integrating honeypots into SCADA systems and critical infrastructure systems in the larger context of securing large-scale networked critical infrastructure. Their honeypot detects attackers by simulating a complete Modbus TCP device, incorporating the protocol, control device logic, as well as other services such as SNMP and FTP services that are commonly found on commercially available PLCs and RTU devices.

3 THE Decepti-SCADA FRAMEWORK

This paper describes the Decepti-SCADA framework, which we believe is a significant improvement over previously available SCADA honeypots. Specifically:

- Decepti-SCADA decoys are built in a modular fashion, making it easy for contributors to develop and add new decoys to the platform. Most existing honeypot system code bases are so coupled that developers have trouble contributing to the projects.
- Most existing honeypot systems are low-interaction, which often compromises deception. Decepti-SCADA's use of Docker helps to create decoys that replicate real operational systems, which makes interactions with decoys highly interactive and thus creates a more convincing deception.

[1] http://www.honeyd.org/
[2] http://conpot.org/
[3] http://scadahoneynet.sourceforge.net/
[4] https://dale-peterson.com/digital-bond-archives/

- Decepti-SCADA's use of Docker eliminates cross-platform dependencies.
- Contra existing honeypot systems, Decepti-SCADA presents a beautiful web graphical user interface for decoy deployment which is even accessible to novice users (See Figures 2 and 5).
- Existing honeypots are generally cumbersome to install, and often take many hours to set up properly. Decepti-SCADA only requires a CentOS 7 minimal installation with git and two network interfaces; `00-startup-script.sh` takes care of the rest.

The purpose of the Decepti-SCADA framework is to trick a network intruder into thinking they're interacting with legitimate SCADA systems; in turn, slowing them down, and at the same time, alerting security analysts of an attacker's presence. There are two main components of the framework, each of which are commonly deployed in a Linux-based virtual machine:

- *Decepti-Box* is used to create, manage, and deploy decoys.
- *ELKSUR* allows network administrators to monitor adversarial interactions with the decoys.

3.1 Decepti-Box and ELKSUR

Decepti-Box is a system deployed on an operational SCADA network which deploys and simulates SCADA devices. Attackers can freely interact with the deployed decoys as if they are real SCADA systems. From a very high level, this is accomplished by a mix of virtual interfaces, and building decoys in docker, such that they are deployed as lightweight containers. Due to the lightweight architecture of Decepti-Box, several hundred of SCADA decoys can be deployed with a minimal resource footprint. The system has a web front end that can be used to easily deploy decoys, even to the most novice of users.

ELKSUR is the second system in the Decepti-SCADA Framework, and is used to detect adversarial interactions with decoys. Like Decepti-Box, ELKSUR is commonly deployed as a virtual machine and specifically monitors TCP/IP and UDP traffic on the same network as Decepti-Box, passively ingesting network traffic and looking for custom made signatures.

Custom signatures are used to detect interactions with the decoys such as: nmap scans, SSH traffic interactions, TCP handshakes, and Banner grabbing. This component is important because once an adversary begins interacting with decoys, thinking that they are real SCADA components, analysts can be alerted in real-time, and thus help mitigate the situation. Currently, there are two ways of ingesting network traffic, inline taps, or port mirroring on a switch.

Inline taps are physical devices which are installed between two network devices such as two routers (Galloway and Hancke, 2012). Port mirroring, also known as a SPAN (Switched Port Analyzer) is accomplished by sending copies of packets seen in a port, to another port, where it can be analyzed (Denning and Denning, 2016) (in our case, to be analyzed by the ELKSUR component of the Decepti-SCADA Framework). The main difference between the two is that inline taps require special devices, and Port Mirroring depends on switch configurations.

3.2 Decepti-SCADA Framework in Action

Traffic is routed into Decepti-box, through a network bridge, into the virtual interfaces, which have SCADA decoy Docker containers bound to them. TCP/IP and UDP traffic can interact with the Containers bound to those internal virtual interfaces. When these interactions happen, a passive IDS is sniffing all traffic, and flags the interactions based on the related network traffic.

4 THE Decepti-SCADA ARCHITECTURE

The Decepti-SCADA framework is broken into two parts as shown in Figure 1, one host, "Decepti-Box" will provide decoy deployment, and the other, "ELKSUR," will detect malicious interactions with the deployed decoys.

Each system has two network interfaces, one plugged into an out-of-band management network, and one on the operational SCADA network. The out-of-band management network includes the front-end web services in which the Security Analyst will use, such as the deployment capabilities. The analyst also uses this network to access metrics on decoy interactions in the form of a web-based Security Information and Event Management (SIEM) system. Each host is heavily dependent on use of docker containers.

4.1 Decepti-SCADA Decoy Templates

Decoy templates are JSON-based, which allows for easy deployments of SCADA-Decoys; they can easily be grouped by categories and platform. This is a new and novel way of deploying containers, as the

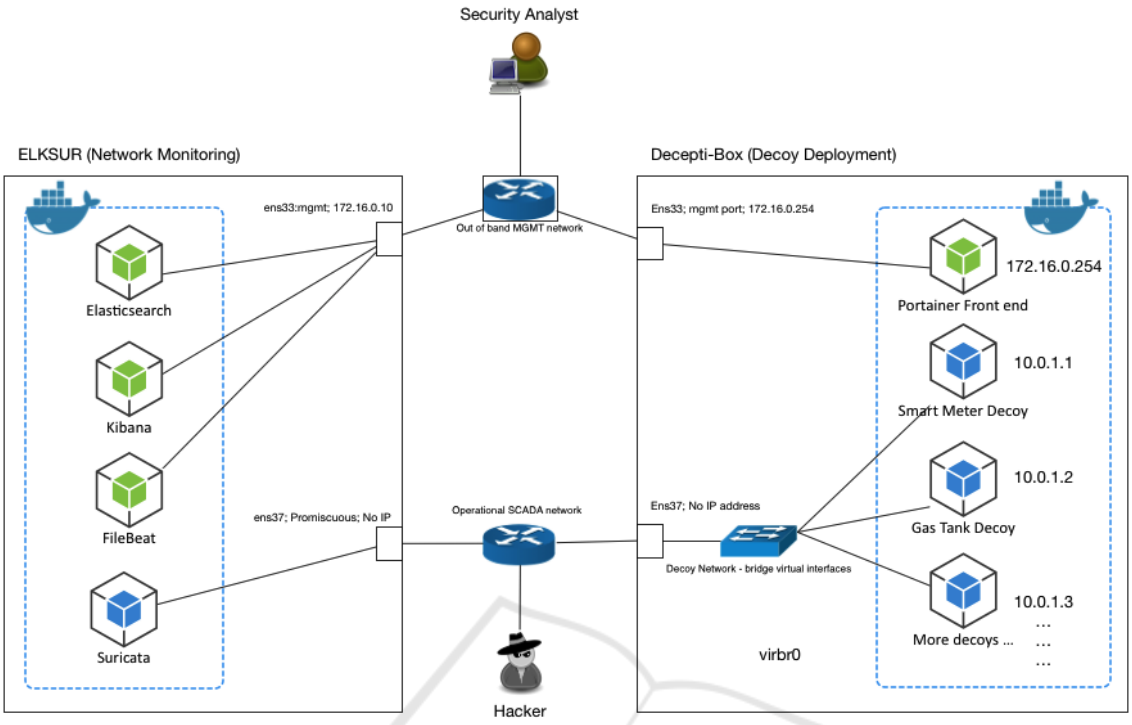

Figure 1: Decepti-SCADA Architecture.

container commands are already packaged for the end user to execute. For example, in a Docker container, a specific command must be executed. With use of a template, we can pre-package that command to be run.

Docker is used as a container runtime for the SCADA decoys. In order to deploy a decoy via docker container, an image of the SCADA device must exist. We have developed SCADA device images including a water meter and Guardian AST gas tank system; however, hundreds of other SCADA decoy Docker images can be created. Docker containers bind to virtual interfaces within Decepti-Box. An example deployment is creating 100 virtual interfaces in Decepti-Box, and assigning virtual IP `10.1.0.1-100` to said interfaces. Docker decoys process and then bind to the virtual IP addresses.

4.2 ELKSUR Components

ELKSUR has four main components:

- *Kibana*[5] is a frontend web GUI to see if anyone has interacted with the decoy. This is the interface in which analysts will use to assess if an intruder is interacting with a SCADA Decoy.

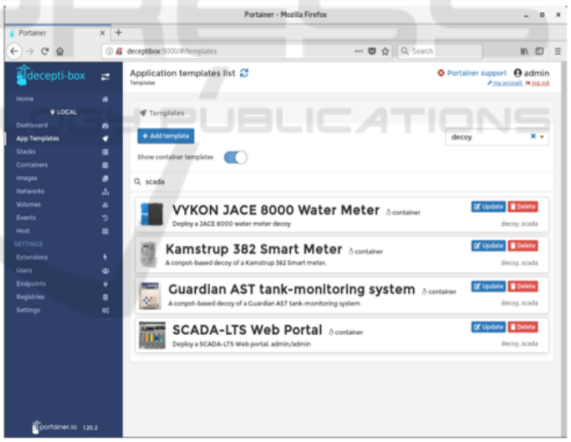

Figure 2: The Decepti-Box Decoy Templates. The Easy-to-Use Decepti-Box User Interface Is Featured.

- *Suricata*[6] is an open source Network-based Intrusion Detection System (IDS). It logs alerts in accordance to signatures; in this case, signatures that pertain to decoy interactions. The log file is called `fast.log`.
- *Filebeat*[7] is a host-based agent used to relay Suricata logs to Elasticsearch. Filebeat relays the suri-

[5]https://www.elastic.co/products/kibana

[6]https://suricata-ids.org/

[7]https://www.elastic.co/products/beats/filebeat

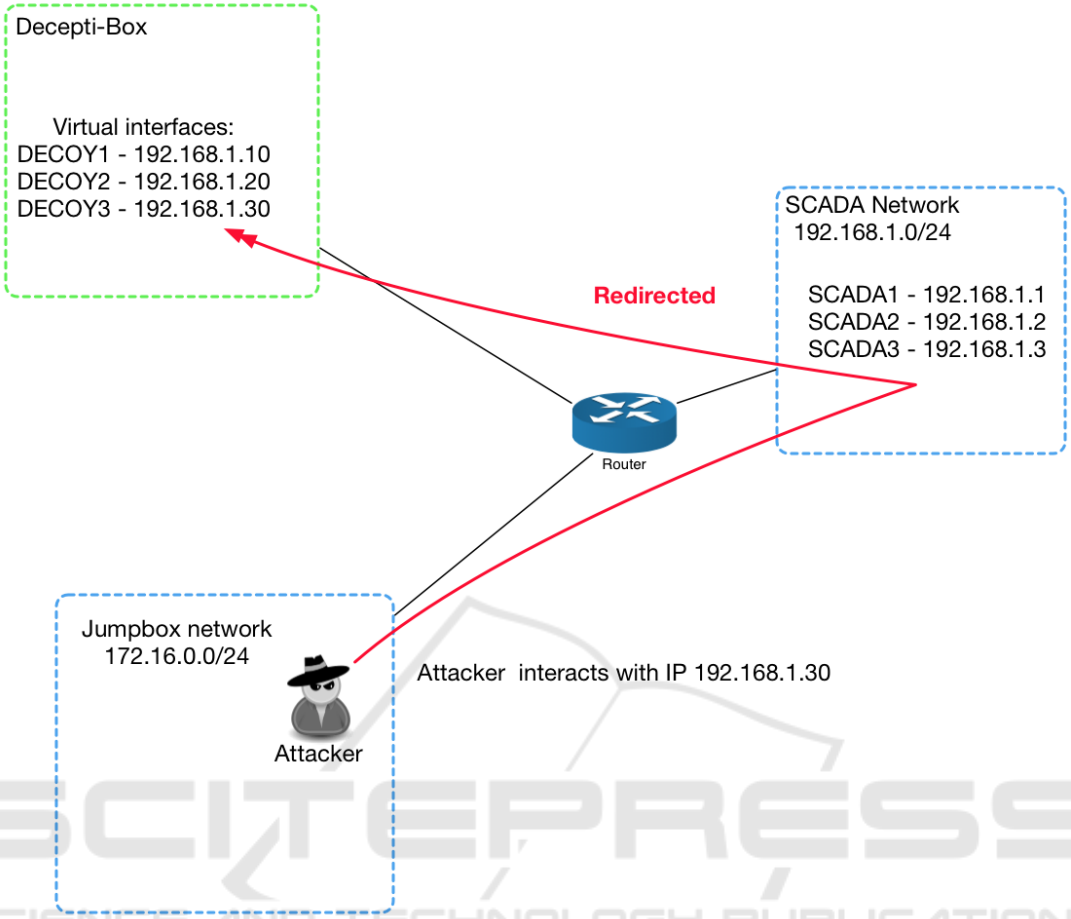

Figure 3: Decepti-SCADA Routing Example.

cata `fast.log` to elasticsearch.

- *Elasticsearch*[8] is a search engine used to collect, and index data received data. Elasticsearch receives the parsed `fast.log`, which it then feeds into the front end, Kibana, for engaging visualizations.

4.3 Routing

Routing traffic into Decepti-Box is environment based; however, traffic must be routed to the virtual IP addresses that reside within Decepti-Box. There are several ways to achieve this, one of which is to redirect traffic from the router.

Figure 3 below depicts an example scenario of how this routing is accomplished: an attacker has breached the Jumpbox network and wishes to interact with a device with a device with the IP address `192.168.1.30`. He believes that this device is located on the SCADA Network subnet; however, when transmitting through the router, it is redirected to Decepti-Box virtual IP address of `192.168.1.30` which houses a SCADA decoy.

Traffic is routed into Decepti-box, through a network bridge, into the virtual interfaces, which have SCADA decoy Docker containers bound to them. TCP/IP and UDP traffic can interact with the Containers bound to those internal virtual interfaces. When these interactions happen, a passive IDS is sniffing all traffic, and flags the interactions based on the related network traffic.

```
docker run -d \
    --name portainer \
    -p 9000:9000 \
    -v /var/run/docker.sock:/var/run/docker.sock\
    -v
/code/portainer_data/templates.json:/templates.json\
    -v /code/portainer_data/data:/data\
    portainer/portainer
```

Figure 4: Initializing Dockerized GUI.

[8]https://www.elastic.co/products/enterprise-search

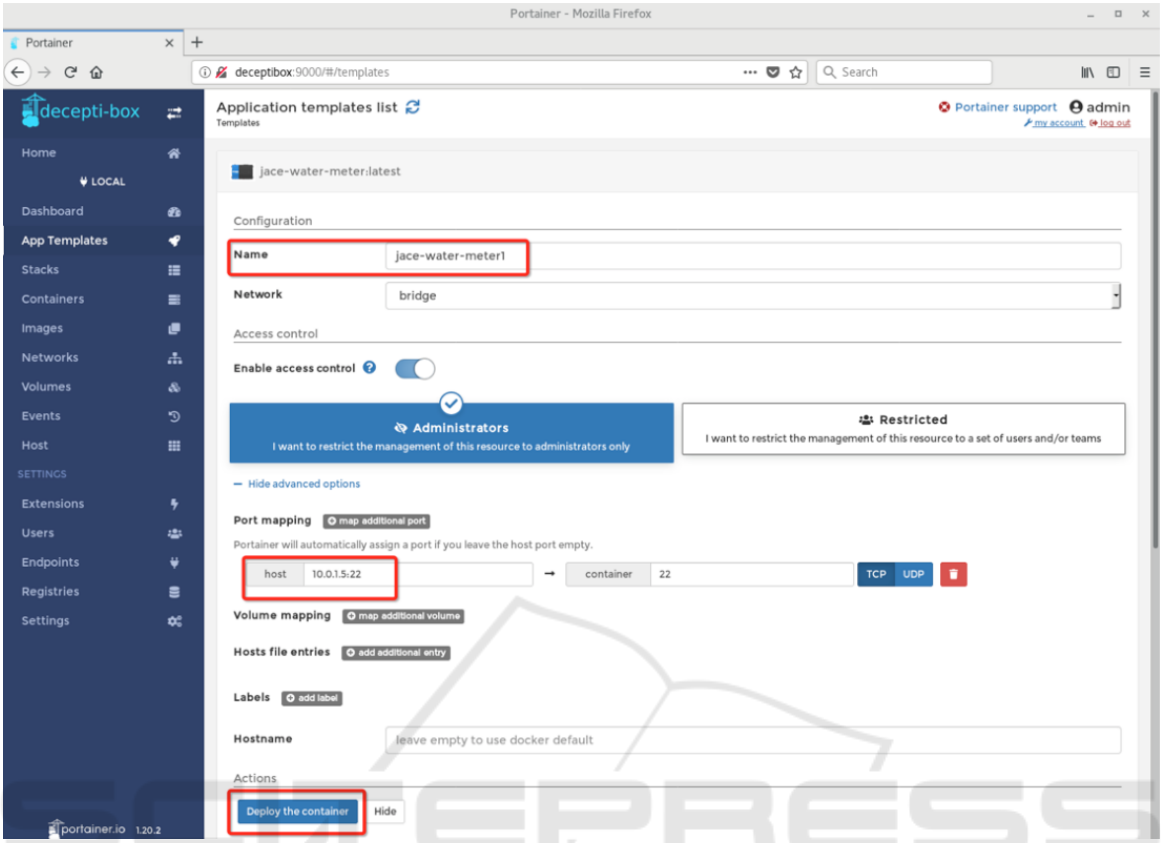

Figure 5: Configuring and Deploying Decoys. A Honeypot Container Is Deployed Masquerading as a JACE Water Meter. Decepti-SCADA Automatically Assigns Ports or Allows Manual Port Assignment. Access Control Can Be Enabled and Limited to Network Administrators or Further Restricted.

5 CONFIGURING AND DEPLOYING A DECOY

The first step in deploying Decepti-SCADA is to start the Dockerized, web-based Graphical User Interface (GUI), called "Portainer." This is accomplished by running the following command:

Once the Portainer GUI is up and running, using a browser, navigate to the IP address, port 9000 of the host on which Portainer is running. To configure and deploy a decoy, from the App Template tab, select the decoy in which you wish to deploy. One option would be to deploy a SCADA Smart Water Meter. To do this, browse, or search for the "VIKON JACE 8000 Water Meter," and select it. Under the "name" field, give it a tasteful description. Next, in the "Host" field, select a virtual IP address to bind the decoy to. Finally, to deploy the decoy, click the "Deploy the Container" button. Reference figure 5 to see this in action.

6 CONCLUSION AND FUTURE WORK

We have introduced and described the Decepti-SCADA framework, a robust solution used to deploy SCADA decoys and continually monitor interactions with them. Decepti-Box is the deployment aspect, while ELKSUR conducts monitoring. While this rapid prototype is not yet an operational system, we have highlighted the ability to deploy SCADA is a user-friendly fashion. Additionally, early results show deception can be accomplished in critical infrastructures by creating realistic decoys of SCADA components.

While the Decepti-SCADA Framework introduced in this paper has shown to be a viable first step in implementing deception across critical infrastructures, and particularly SCADA components. We are continually making improvements to the Decepti-SCADA prototype:

- We are currently working on a Decepti-Box build for a Windows client.
- We will be utilizing machine learning and artificial intelligence capabilities to enable decoys to be more active and dynamic thus creating improved realism.
- We continue to profile various equipment components from multiple vendors in order to create a more diverse set of decoys.
- We are actively studying ways to create more decoys while maintaining system fidelity.
- We are beginning to study game theoretic and logical tools to more effectively camouflage Decepti-SCADA honeypots among legitimate SCADA network assets. This might be accomplished through the implementation of deceptive network scan results (Jajodia et al., 2017).
- We are conducting is an implementation of an attacker engagement strategy (Bilinski et al., 2019) in a SCADA network environment. The experiment would have a real device and a decoy on a SCADA network and test to see how a Reinforcement Learning (RL) agent would be perform in determining which of the two devices are real. Instead of using a simulated environment to give signals to the RL agent, our experiment would use real signals from the real machine and decoys. This experiment will give us insight on how our decoy system will influence an attacker given they are aware of the techniques deployed by Decepti-SCADA.

Beyond the areas described above, the Decepti-SCADA Team continues to work on more refined testing of the various components as well as develop a case study which involves red team involvement in order to determine the utility of deception for SCADA.

ACKNOWLEDGEMENTS

Roger A. Hallman is supported by the United States Department of Defense SMART Scholarship for Service Program funded by USD/R&E (The Under Secretary of Defense-Research and Engineering), National Defense Education Program (NDEP) / BA-1, Basic Research.

REFERENCES

AlSabah, M. and Goldberg, I. (2016). Performance and security improvements for tor: A survey. *ACM Computing Surveys (CSUR)*, 49(2):32.

Araujo, F., Hamlen, K. W., Biedermann, S., and Katzenbeisser, S. (2014). From patches to honey-patches: Lightweight attacker misdirection, deception, and disinformation. In *Proceedings of the 2014 ACM SIGSAC conference on computer and communications security*, pages 942–953. ACM.

Arghira, N., Hossu, D., Fagarasan, I., Iliescu, S. S., and Costianu, D. R. (2011). Modern scada philosophy in power system operation-a survey. *University" Politehnica" of Bucharest Scientific Bulletin, Series C: Electrical Engineering*, 73(2):153–166.

Bilinski, M., Ferguson-Walter, K., Fugate, S., Gabrys, R., Mauger, J., and Souza, B. (2019). You only lie twice: A multi-round cyber deception game of questionable veracity. In *International Conference on Decision and Game Theory for Security*, pages 65–84. Springer.

Buller, D. B. and Burgoon, J. K. (1996). Interpersonal deception theory. *Communication theory*, 6(3):203–242.

Chakraborty, T., Jajodia, S., Katz, J., Picariello, A., Sperli, G., and Subrahmanian, V. (2019). Forge: A fake online repository generation engine for cyber deception. *IEEE Transactions on Dependable and Secure Computing*.

Denning, D. E. (2014). Framework and principles for active cyber defense. *Computers & Security*, 40:108–113.

Denning, P. J. and Denning, D. E. (2016). Cybersecurity is harder than building bridges. *American Scientist*, 104(3):155.

Dwork, C. (2011). Differential privacy. *Encyclopedia of Cryptography and Security*, pages 338–340.

Fan, W., Du, Z., Fernández, D., and Villagrá, V. A. (2017). Enabling an anatomic view to investigate honeypot systems: A survey. *IEEE Systems Journal*, 12(4):3906–3919.

Galloway, B. and Hancke, G. P. (2012). Introduction to industrial control networks. *IEEE Communications surveys & tutorials*, 15(2):860–880.

Gutzwiller, R., Ferguson-Walter, K., Fugate, S., and Rogers, A. (2018). "oh, look, a butterfly!" a framework for distracting attackers to improve cyber defense. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 62, pages 272–276. SAGE Publications Sage CA: Los Angeles, CA.

Jajodia, S., Ghosh, A. K., Subrahmanian, V., Swarup, V., Wang, C., and Wang, X. S. (2012). *Moving Target Defense II: Application of Game Theory and Adversarial Modeling*, volume 100. Springer.

Jajodia, S., Park, N., Pierazzi, F., Pugliese, A., Serra, E., Simari, G. I., and Subrahmanian, V. (2017). A probabilistic logic of cyber deception. *IEEE Transactions on Information Forensics and Security*, 12(11):2532–2544.

Latimer, J. (2003). *Deception in War: Art Bluff Value Deceit Most Thrilling Episodes Cunning mil hist from The Trojan*. Abrams.

Merkel, D. (2014). Docker: lightweight linux containers for consistent development and deployment. *Linux journal*, 2014(239):2.

Nazir, S., Patel, S., and Patel, D. (2017). Assessing and augmenting scada cyber security: A survey of techniques. *Computers & Security*, 70:436–454.

Palomäki, J., Yan, J., and Laakasuo, M. (2016). Machiavelli as a poker mate—a naturalistic behavioural study on strategic deception. *Personality and Individual Differences*, 98:266–271.

Pawlick, J., Colbert, E., and Zhu, Q. (2019). A game-theoretic taxonomy and survey of defensive deception for cybersecurity and privacy. *ACM Computing Surveys (CSUR)*, 52(4):82.

Romero-Mariona., J., Hallman, R. A., Kline., M., Miguel., J. S., Major., M., and Kerr., L. (2016). Security in the industrial internet of things - the c-sec approach. In *Proceedings of the International Conference on Internet of Things and Big Data - Volume 1: IoTBD,*, pages 421–428. INSTICC, SciTePress.

Simões, P., Cruz, T., Gomes, J., and Monteiro, E. (2013). On the use of honeypots for detecting cyber attacks on industrial control networks. In *Proc. 12th Eur. Conf. Inform. Warfare Secur. ECIW 2013*.

Simões, P., Cruz, T., Proença, J., and Monteiro, E. (2015). Specialized honeypots for scada systems. In *Cyber Security: Analytics, Technology and Automation*, pages 251–269. Springer.

Wheatley, D. (1976). Deception in world war ii. *The RUSI Journal*, 121(3):87–88.