Taking Control: Design and Implementation of Botnets for Cyber-Physical Attacks with CPSBot

Daniele Antonioli
Singapore University of Technology and Design
daniele_antonioli@mymail.sutd.edu.sg

Giuseppe Bernieri
Roma Tre University
giuseppe.bernieri@uniroma3.it

Nils Ole Tippenhauer
Singapore University of Technology and Design
nils_tippenhauer@sutd.edu.sg

ABSTRACT

Recently, botnets such as Mirai and Persirai targeted IoT devices on a large scale. We consider attacks by botnets on cyber-physical systems (CPS), which require advanced capabilities such as controlling the physical processes in real-time. Traditional botnets are not suitable for this goal mainly because they lack process control capabilities, are not optimized for low latency communication, and bots generally do not leverage local resources. We argue that such attacks would require *cyber-physical* botnets. A cyber-physical botnet needs coordinated and heterogeneous bots, capable of performing adversarial control strategies while subject to the constraints of the target CPS.

In this work, we present CPSBot, a framework to build cyber-physical botnets. We present an example of a centralized CPSBot targeting a centrally controlled system and a decentralized CPSBot targeting a system distributed control. We implemented the former CPSBot using MQTT for the C&C channel and Modbus/TCP as the target network protocol and we used it to launch several attacks on real and simulated Water Distribution. We evaluate our implementation with *distributed reply* and *distributed impersonation* attacks on a CPS, and show that malicious control with negligible latency is possible.

KEYWORDS

Botnets, Attacks, CPS, ICS, IoT, BAS, Control, MQTT, Modbus

1 INTRODUCTION

Botnets are still one of the major threats in the cyber-security landscape. IT botnets take advantage of Internet services such as IRC, HTTP, email, and DNS to achieve different goals including information and identity theft, spam, DDoS and malware distribution [48]. The Mirai and Persirai IoT botnets are typical examples of IT botnets for DDoS [6]. In the OT space, we have seen advanced malware such as Stuxnet [18] and Blackenergy [29, 36] using botnet-like components to affect the availability of the target cyber-physical system (e. g., DoS and damaging the equipment).

However, we have not seen botnets capable of cyber-physical system (CPS) *adversarial control*. We claim that current botnet designs are not sufficient to achieve this goal, mainly because high-impact attacks on a CPS require different strategies than conventional cyber-security attacks [23, 49]. These strategies translate into additional requirements that are not addressed by conventional botnet designs. For example, a conventional botnet does not differentiate its bots according to the capabilities of the infected devices and does not allow coordinated interactions among the bots. We think that it is beneficial to introduce a new class of botnets, defined as *cyber-physical* botnets, designed to overcome those additional challenges. We expect that better understanding of capabilities and shortcomings of cyber-physical botnets will raise awareness with stakeholders of threatened systems, and allow the defenders to design more suitable countermeasures.

In this paper, we present **CPSBot**, a framework to build cyber-physical botnets. CPSBot enables to build botnets with heterogeneous and coordinated bots able to take over the control of a CPS. CPSBot is generic over the target CPS, it allows to develop botnets with different network architectures and to use different adversarial control strategies. We underline two of the most important design choices that we made to satisfy our requirements. Firstly, we use a novel command and control channel based on the *publisher-subscriber (PubSub)* paradigm [7, 17] to get precise coordination among bots with minimal overheads. Secondly, we define a set of orthogonal functionality that we call *traits* to customize the development of a CPSBot. This modular approach is used to exploit the functionalities offered by different infected devices of a cyber-physical system and to customize the C&C servers.

Our attacker model considers a botmaster that already managed to infect the target devices (how is outside the scope of this work). We present two design examples of CPSBots targeting a centrally controlled system and a system with distributed control. We compare our examples against the traditional counterparts and we motivate why we think that the latter are not sufficient to enable adversarial control of the target systems. We provide an implementation of the centralized botnet where an attacker is coordinating infected gateway devices to influence the distribution of water across remote substations. We use MQTT for the C&C protocol and we target the Modbus/TCP industrial protocol.

We used our implementation to perform two coordinated cyber-physical attacks on real and simulated Water Distribution. The first attack is defined as *distributed impersonation* and the attacker is able to simultaneously impersonate geographically-sparse remote terminal units. The second attack is defined as *distributed replay* and the attacker is able to reply values and control actions across (potentially heterogeneous) devices in different substations.

We argue that traditional botnet metrics, such as number of bots and DoS bandwidth, are not sufficient to evaluate a cyber-physical botnet. Hence, we define our own set of quantitative metrics suitable to evaluate the CPSBot framework such as adversarial control period and additional delay introduced by CPSBot and we use those metrics to evaluate our cyber-physical attacks.

We summarize our contributions as follows:

- We propose *CPSBot* a framework to design cyber-physical botnets. Our framework addresses the extra-requirements introduced when attacking a cyber-physical system such as adversarial control and bots coordination capabilities. We use a publisher-subscriber command and control channel and traits to address those extra-requirements.
- We design a centralized and a decentralized CPSBots targeting a centrally controlled system and a system with distributed control. We implement the former botnet optimizing it for precise coordination among bots using MQTT features such as quality of service, persistent sessions and asynchronous communication.
- We launch two coordinated cyber-physical attacks: distributed impersonation and distributed replay to assess our implementation. Both attacks were performed on simulated and real water distribution testbeds, with minor code modifications. We evaluate them with our quantitative metrics for cyber-physical botnets.

This work is organized as follows: in Section 2, we provide the background about botnets, cyber-physical systems and our target water distribution system. In Section 3 we present the design of CPSBot starting from our problem statement and threat model. We focus on the PubSub C&C channel and the CPSBot traits. Then, we show two examples of centralized and decentralized CPSBots. We conclude the section with our set of quantitative metrics for cyber-physical botnets. In Section 4, we present how we implemented a centralized CPSBot using MQTT and Modbus/TCP to attack our target water distribution system. In Section 5 we describe the CPSBot attack phases and we present and evaluate the distributed impersonation and distributed replay attacks using the implemented centralized botnet. We discuss several CPSBot attack strategies and optimizations in Section 6. Related works are summarized in Section 7, and we conclude the paper in Section 8.

2 BACKGROUND

2.1 Botnets

A botnet is a network of compromised hosts (bots) that are managed by one or more command and control (C&C) servers. The attacker (botmaster) is connected to the C&C infrastructure and she is sending directives to the bots through it. The channel of communication between the attacker and the bots is called *C&C channel*, and it is one of the most important parts of a botnet [48]. A canonical way to classify a botnet is by its network architecture. Most commonly a botnet is either *centralized* or *decentralized*[1].

A centralized architecture has one C&C server that communicates with all the bots. A client-server protocol is used for the C&C channel (e. g., IRC or HTTP). The main advantages of this setup are its low latency and ease of coordination. The main weaknesses of this setup are its vulnerability to single point of failure and network scalability issues when the number of bots increases. Alternatively, in a decentralized architecture, all compromised devices are used both as bots and C&C servers. The C&C channel uses a peer-to-peer protocol (P2P) such that the bots establish an overlay network. This architecture is self-scalable and does not suffer from single point of failure. However, it might be difficult to implement (e. g., hosts behind NAT) and coordinate (e. g., orders from multiple C&C). There are also *hybrid* architectures that provide a tradeoff between centralized and decentralized schemes, and *random* architectures where the bots are not contacting the C&C server but they are waiting to be contacted by the botmaster.

It is possible to represent the state of a botnet using a five-phases *lifecycle*. We have an *initial infection* phase, where the attacker exploits one or more vulnerabilities on a remote machine. The remote machine becomes a bot candidate. In the *second infection* phase, the same infected machine is instructed to download and execute different types of malware. If the malicious code is effective then the infected machine becomes part of the botnet. In the third phase, the bot contacts the C&C server and this process is defined as *rallying*. The rallying phase might be accomplished using static addresses or dynamic addressing techniques such as DNS fast-flux, and domain generation algorithms (DGA). The fourth phase is the *attack* phase, where the bot perform malicious activities and might still exchange information with the C&C (e. g., exfiltrate data). The last phase is the *maintenance* phase, where a botmaster might modify the bot network configuration, upload new attack payloads and update the cryptographic keys of the botnet.

The C&C channel uses either the same protocol of the target system or a custom protocol (defined as neoteric [55]). The former approach is, in general, less easy to detect because the malicious network traffic is similar to the expected one. However, once the extra-traffic is detected then the defender can isolate the offending traffic and try to understand what is going on. The latter approach generates network traffic that might stand out compared to the normal one. On the other hand, it is more difficult for the defender to decode the information carried in a neoteric packet [28].

2.2 Cyber-Physical Systems (Security)

Cyber-Physical Systems (CPS) are composed of heterogeneous devices that are interacting with a physical process. These devices are typically interconnected and they are programmed to perform general-purpose or domain-specific tasks, including sensing, actuating, and networking. It is possible to divide cyber-physical systems (CPS) in two categories: CPS with a *central* or *distributed* control. In the first case, the CPS uses a central monitoring and control infrastructure. For example, a water distribution industrial control system is centrally controlled by a SCADA server. In the second case, the CPS uses multiple controllers, each controller manages a sub-system and it is able to communicate with the other controllers. For example, a building automation system (BAS) is an example of a distributed control system. There are also examples of distributed control system where the control logic of one controller depends on signals coming from other ones. These signals are called interlocks. A water treatment system is an example of an interlock-based distributed control system.

The usage of Internet-friendly protocols and commodity hardware vastly increased the attack surface of cyber-physical systems [9, 30, 32, 38]. CPS are vulnerable to classic information security threats, attacks targeting the underlying physical processes, and the intersection between the two [24]. In this paper, we are interested in the latter type of attacks, defined as *cyber-physical*

[1] In this context decentralized in synonymous of distributed.

attacks (e. g., attack over the network that permanently damages a local component).

2.3 The Water Distribution (WaDi)

The Water Distribution (WaDi) testbed is a water distribution autonomous systems built at Singapore University of Technology and Design (SUTD) in 2016. It is designed as a down-scaled version of the water transmission and distribution system operated in Singapore. WaDi enables simulating different water demand patterns, water hammer effects, changes in pipe pressures and pipe leakages. Furthermore, it allows simulating water pollution by means of organic and inorganic contaminants that can be added to the distribution of water to the consumers.

WaDi is composed of three sub-processes: *water supply*, *water distribution*, and *water return*. The first stage operates by taking the source water from two elevated raw water tanks (2500L each) and transferring it to two elevated reservoirs (1250L each). In this stage, water quality analyzers are used to verify the incoming water quality. In the second stage, the potable water is distributed to six consumers tanks (500L each). The water demand of Each consumer tank can be set independently and changed in real-time. In the third stage, the water is collected in the single water return tank (2000L) and then optionally returned back to the supply stage.

WaDi has different types of sensors (e. g., water level sensors, flow meters, water quality sensors) and actuators (e. g., water pumps, valves). The remote terminal units are SCADAPack 334E devices (Schneider Electric), the gateway devices are MOXA oncell G3111-HSPA and the industrial switches are MOXA ED5 205A. The WaDi supervisory network uses the Modbus/TCP industrial protocol.

3 CPSBOTS FOR CYBER-PHYSICAL ATTACKS

In this section, we present our problem statement and the related system and attacker model. Then, we focus on the CPSBot C&C channel and on the traits. We show the high-level architecture of two CPSBots: a centralized botnet for ICS and a decentralized botnet for IoT. We conclude the section defining a set of quantitative metrics derived from the design of our botnets.

3.1 Problem Statement

Our main challenge is related to *adversarial control* of a system that has a physical process and several interconnected heterogeneous devices (i. e., a cyber-physical system). With adversarial control, we refer to the attacker's capability to steer the physical process of the target system into states of her choice. Typically, this requires both understanding of the physical process, and a suitable control strategy (e.g., using closed loop control).

We think that traditional botnets are not sufficient to launch coordinated attacks on CPS systems for several reasons. Firstly, they do not address the problem of adversarial control, and they are instead focusing mostly on system disruption (e. g., DDoS) and observation (e. g., eavesdropping). Adversarial control of the system would require near real-time exchange of eavesdropped data and malicious control commands between distributed bots together with one or more adversarial controllers. We argue that traditional botnets are not designed to provide such features. Secondly, IT botnets consider their bots as a homogeneous set of devices (e. g., zombies), without exploiting their different hardware and software capabilities [48]. However, in a cyber-physical botnet, some bots have different roles such as influencing the physical process and spoofing the monitoring system. Thirdly, traditional botnet designs are not optimized for CPS constraints such as latency, packet size and throughput. In our case, these constraints have to be taken into account and measured in some way.

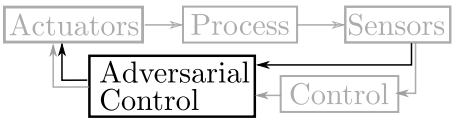


Figure 1: Attacker goal: real-time adversarial control strategy using a cyber-physical botnet.

In summary, we would like to build a new class of botnets able to perform coordinated cyber-physical attacks. We define those botnets as *cyber-physical*. In general, a cyber-physical botnet have to resemble a control system, and the main difficulties to be addressed in its design are the following:

- Introduce an adversarial and coordinated control strategy on a CPS in real-time.
- Exploit the diversity of the bots, including their specific hardware and software capabilities.
- Evaluate the tradeoff between richness of bots functionality and associated overheads using sound metrics.

3.2 System and Attacker Models

Our system model focuses on two types of cyber-physical systems: a system with centralized control and a system with distributed control.

Centralized Control. We consider a (potentially geographically distributed) system composed of n substations that are centrally controlled (e. g., a water distribution system). Each substation is controlled by a remote terminal unit (RTU) that is able to read sensor values and control actuators. There are no intrusion detection systems deployed in the substations. A central SCADA server is periodically monitoring the substations and it can send commands to the RTUs. A network-based intrusion detection system (IDS) is monitoring the inbound and outbound traffic from the SCADA network.

Distributed Control. We consider an IoT system with distributed control (e. g., a building automation system) that is deploying m devices. Each device has a specific functionality such as monitor and control the temperature of a room (e. g., HVAC), video surveillance (e. g., CCTC), and lighting control system (e. g., LCS).

Attacker model. We consider an attacker who already completed the necessary steps to map the network of the target system, infect the devices, and perform the rallying phase (e. g., the attacker is able to remotely contact the bots via the C&C). We believe that this is a reasonable attacker model to adopt both in the ICS and IoT scenario given the recent trends and surveys [15, 37, 43, 46]. The main goal of the attacker is to use bots in a coordinated fashion to take control over the target cyber-physical system, adding an adversarial operation in the closed-loop control routines (see Figure 1). In this

setup the attacker, as the original controller, takes advantage of readings from multiple sensors (coming from different bots) and use this knowledge to send malicious actuation command to drive the system to an arbitrary state or sequence of states. Throughout the paper, we use the words attacker and botmaster interchangeably.

3.3 CPSBot: PubSub C&C Channel

We propose to use *publish-subscribe (PubSub)* messaging pattern for the C&C channel. In a PubSub scheme, there are three entities: the publisher (sender), the subscriber (receiver) and the broker (dispatcher). The communication is *event-driven* and there is a loose coupling between the sender and the receiver. This scheme encourages the use of asynchronous communications in contrast with client-server periodic request-response cycles (e. g., polling) [7, 17]. In our context, we consider a cyber-physical botnet with heterogeneous bots subscribed to relevant events (e. g., sensor values and actuator states) and one or more C&C nodes publishing commands and updating events based on those values in real-time. To the best of our knowledge, the PubSub scheme has not been proposed for a botnet C&C channel before, and is particularly well suited for our application.

Traditional IT botnets are using different C&C control protocols mainly because they target different systems (e. g., client-server architecture). For example, IRC, DNS, email, and HTTP protocols are popular choices [55]. In our system models (e. g., ICS, and IoT) those choices are either inapplicable (e. g., the protocol is not spoken in the system) or sub-optimal. As an illustration, HTTP is sub-optimal because it is a client-server protocol (not event-driven) and it is not designed for machine-to-machine communications (packet size is not a problem)[2].

We list some crucial advantages that we think we would gain from a PubSub control channel compared to conventional ones in the context of ICS and IoT systems:

- Flexible coordination among bots and C&C :
  - Enabled by event-driven messaging.
  - It allows using synchronous and asynchronous messaging schemes, multicast traffic, proactive and reactive bots coordination.
- Compatibility with different botnet architectures:
  - Enabled by loose coupling of publishers and subscribers.
  - It allows building conventional (centralized, decentralized) and non-conventional (hybrid, random) botnets.
- Addresses traditional and CPS botnets constraints:
  - Enabled by the nature of PubSub intended for reliable and secure machine-to-machine communication [10, 41].
  - It allows scaling the botnet size maintaining low computational and traffic overheads. For example, we can use anonymity [16], confidentiality and integrity mechanisms such as TLS or alternatives [27].

3.4 CPSBot: Bot Traits

We already introduced the problem of cyber-physical bot heterogeneity that translates into the need of bots supporting different functionalities. Those functionalities derive from the role of a bot in an attack and they are limited from the software and hardware capabilities of the bot. We address the heterogeneity challenge using CPSBot *traits*. A trait represents a set of functionalities that a CPSBot device might support. This enables to design a cyber-physical botnet that is modular (e. g., reuse same functionality across different devices), extensible (e. g., improve a functionality without affecting the others), and composable (e. g., mix functionalities in a single device). We note that traits allow to customize *both* the bots and the C&Cs. We borrow the concept of trait from object-oriented programming theory [47]. We describe six traits that are relevant to our paper:

Infiltrator. The Infiltrator affects the network configuration of the infected device. For example, it might configure the bot as a malicious proxy able to passively observe traffic, actively send payloads, forward traffic to another network interface, and disconnect the bot from an arbitrary network. In a typical setup, the number of Infiltrators scales linearly with the number of CPSBot bots.

Forger. The Forger tampers with the data coming from sensors, actuators, and other connected devices. For example, it might locally modify an actuator value while spoofing a remote monitoring server. We note that the Forger takes advantage of different hardware capabilities of the infected bots (e. g., influence the physical process). In a typical setup, the number of Forgers is proportional to the number of CPSBot bots.

Controller. The Controller takes care of the adversarial control of the target system. In general, the Controller takes input from the cyber-physical system and optionally from other CPSBot bots, and predicts the future input-output state. The prediction could be computed using different orthogonal techniques such as machine-learning classification, real-time simulation, and state estimation techniques (e. g., Kalman filtering). This functionality is typically implemented by the C&C in a centralized CPSBot, and by the infected controller devices in a decentralized CPSBot. A discussion about different prediction strategies is presented in Section 6.1.

Broker. The Broker functionality is used to coordinate the CPSBot network. It asynchronously and synchronously sends event information to all the botnet nodes. For example, a Broker manages the communication between two bots without having them know each other and even if one of them is disconnected from the CPSBot network. Typically, this functionality is implemented by the C&C in a centralized botnet and by multiple nodes (e. g., broker clustering) in a decentralized botnet. We note that, an architecture with multiple Brokers tolerates single-point-of-failure in the CPSBot network.

Pub. The Pub allows the CPSBot devices to send data over the botnet network asynchronously. The granularity of the published content can be set (e. g., publish an aggregate of sensor values versus a single sensor value). Furthermore, the Pub could set the quality of service of each published (sent) value. For example, the botmaster can coordinate the botnet using an event-based priority scheme dependent by the message type (topic-based) or the message value (content-based). This functionality is typically carried out by all CPSBot nodes.

Sub. The Sub allows the CPSBot devices to receive data over the CPSBot network. Each Sub might subscribe to any information exchanged in the CPSBot network, and get it on-demand, without sending a request all the times (event-driven). The subscription

[2] We can apply a similar reasoning for the other traditional C&C protocols.

process is pre-configurable to avoid re-subscriptions after successive disconnections. Additionally, each Sub can create a session with the Broker to let it cache lost messages and retrieve them among re-connection. For example, the C&C node might subscribe to status-critical information to be informed when any of the bots is disconnected from the CPSBot network and react accordingly. Similarly to the Pub functionality, the Sub functionality is typically implemented by all CPSBot nodes.

In summary, we think that designing a botnet using traits is an effective way to address the diversity of devices found in cyber-physical systems. For example, we can use traits to differentiate the implementations of bots for network spoofing and bots for adversarial control (we will see two concrete examples in Section 3.5). As a side benefit, the usage of traits lowers the development costs of our CPSBots and this is a key factor for effective cyber-physical attacks [23].

3.5 Centralized and Decentralized CPSBots

We now present two CPSBots: a centralized botnet attacking a water distribution system (ICS) and a decentralized botnet attacking a building automation system (IoT). We choose these examples because they share similar security functions and weaknesses [8]. For the sub-figures in Figure 2, we represent the devices controlled by the attacker with black squared boxes. The traits of the CPSBot devices are represented as black boxes with round corners. The grey boxes represent the targets. Both botnets derive from the system and attacker models presented in Section 3.2.

Centralized CPSBot. In this scenario, the attacker is using a centralized architecture to attack a centralized control system (e. g., water distribution system). As depicted in Figure 2a, the system is composed of n remote substations and a central monitoring SCADA network. Each substation has a remote terminal unit (RTU) that interacts with sensors and actuators and a gateway device that connects the RTU to the access router. The botmaster has compromised n gateway devices ($\text{Bot}_1, \ldots, \text{Bot}_n$). Each bot implements the Forger, Infiltrator, Pub and Sub traits. The central C&C is managing all the bots and it is implementing the Controller, Broker, Pub, and Sub traits. The bots are altering the state of the physical process in real time with the help of the C&C while fooling the central SCADA server that is periodically querying the RTUs. This CPSBot design is different from a traditional one because each bot acts in a different manner according to its substation. For example, the first bot is mainly interested in the sensor and actuator values regarding the first substation and it generates spoofed commands accordingly. In a traditional setup, each bot will execute the same orders from the C&C regardless of which substation it is affecting. More information about an implementation of this botnet are presented in Section 4 and the two cyber-physical attacks are evaluated in Section 5.

Decentralized CPSBot. In this case, the botmaster is using a decentralized architecture to attack a distributed control system, in particular, a building automation system (BAC). As we can see from Figure 2b, the BAC is composed of a heating, ventilation and air-conditioning system (HVAC), an IP camera (CCTV) and a lighting control system (LCS). In this case, each infected device acts both as a bot and as a C&C. All the bots are implementing the Controller trait and they are locally computing their adversarial estimations. Additionally, they implement the Infiltrator, Forger, Sub and Pub traits. In this scenario, we take advantage of multiple brokers (e. g., every bot implements the Broker trait) to avoid single-point-of-failure. For example, if one bot is compromised the others can still coordinate their actions. This botnet is different from a traditional P2P botnet because each bot has a specific control strategy depending on the infected device. Additionally, each bot might share control information with the others even if the target system is not interlock-based. For example, a botmaster could use *adversarial interlocks* to perform cascade cyber-physical attacks (e. g., induce an LCS blackout while tampering with the HVAC load).

Table 1: Quantitative metrics used to evaluate the CPSBot attacks. A checkmark (✓) in the CPS column indicates that the metric is defined by us to address the CPSBot cyber-physical constraints.

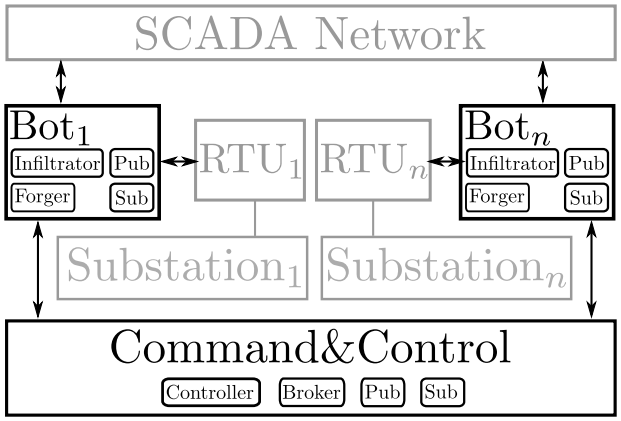
Symbol	Metric description	CPS
T_S [s]	Period between two equal requests from the defender.	✓
T_C [s]	Adversarial estimation period.	✓
T_M [s]	Traffic manipulation period.	✓
Δ_s [ms]	Delay between the last valid packet and the first spoofed packet.	✓
Δ_r [ms]	Delay between the last valid request and the first spoofed response.	✓
μ_d [ms]	Additional delay introduced by the CPSBot.	✓
n_B	Number of bots.	✗
n_e	Number of IDS warnings and errors raised.	✗
μ_{CPU}	Average CPU overhead for the bot.	✗
μ_{RAM}	Average RAM overhead for the bot.	✗

3.6 Our Quantitative Metrics

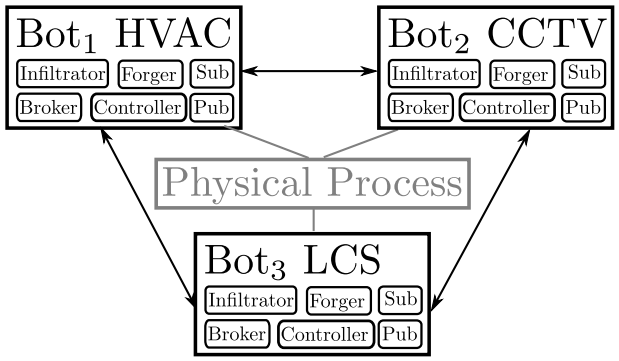
Conventional botnets are evaluated looking at factors such as number and geo-locations of bots and C&C servers, malicious DNS activities, and malware databases [35, 51]. However, we think that to evaluate cyber-physical botnets we have to define additional metrics. Table 1 lists our set of quantitative metrics that we are using in Section 5.5 to evaluate our cyber-physical attacks. A checkmark (✓) in the CPS column indicates that the metric is defined by us to address the CPSBot cyber-physical constraints.

4 DESIGN AND IMPLEMENTATION FOR ICS

In this section, we present an implementation of a centralized CPSBot to attack a water distribution industrial control system. In particular, the system is controlled using the Modbus/TCP protocol. We start by mapping the network architecture sketched in Figure 2a to a water distribution network. We then describe how we deal with the target industrial protocol and the botnet C&C channel protocol. We conclude the section showing how we implemented some specific PubSub features to better coordinate the CPSBot bots.



(a) Centralized botnet targeting a distributed ICS (e. g., water distribution system).



(b) Decentralized botnet targeting an IoT system (e. g., building automation system)

Figure 2: High-level view of an ICS and IoT CPSBot. Grey boxes represent the targets. The black boxes are the devices controlled by the attacker. Rounded black boxes represent the traits supported by the CPSBot devices.

4.1 Network Topology

Figure 3 shows the network topology of a water distribution ICS already compromised by a centralized CPSBot. There are $n + 2$ networks: n substation networks, the SCADA network, and the attacker network. Each network has a border router connected to the Internet. The remote terminal units (RTU) are managing local sensors and actuators and they are communicating with the central SCADA server through gateway devices. A network-based intrusion detection system (IDS) is monitoring the inbound and outbound traffic from the SCADA network. The attacker infected n gateway devices that are sitting in between the RTUs and the border routers. The botmaster uses the symbolic links colored in red to communicate with the bots.

4.2 Target Protocol: Modbus/TCP

We choose Modbus [33] and in particular Modbus/TCP as a target industrial protocol because it is still widely used on actual industrial plants[39]. Furthermore, Modbus/TCP is adopted in WaDi, the water distribution testbed that we use for our attacks. We understand that Modbus (as many popular industrial protocols) is not secure by design. However, we are not interested in discovering or underlying existent Modbus vulnerabilities.

Modbus includes two data types: registers and bit fields. Registers are 16-bit integers and they are either read-only (input registers) or read and write (holding registers). Bit fields are either read-only (discrete inputs) or read and write (coils). All data types are addressed like an array in memory and the first array element is at offset zero. Modbus/TCP is a client-server application layer protocol. A Modbus request has a Modbus/TCP header that contains: the transaction number (set by the client and echoed back by the server), the length of the payload, the protocol ID, and the slave ID. The payload of the Modbus request addresses the requested data with a function code, a memory offset, and a word count. A Modbus response contains a similar Modbus/TCP header and its payload contains the same function code of the corresponding request, a byte count, and the requested data.

Modbus operations are encoded with numeric function codes. The attack that we present in Section 5 focus on three of them: read coils (`0x01`), read holding registers (`0x03`), and write single coil (`0x05`). Read coils is used by a Modbus client to read multiple binary values from a Modbus server. Read holding registers are used by a Modbus client to read multiple 16-bit registers from a Modbus server. Write single coil is used by the Modbus client to write a single bit to a Modbus server.

It is worth mentioning that the Modbus packets are transmitted using network ordering (e. g., big endian) but the bit fields are stored in reverse bit order. Hence, if the first byte of the coil memory contains the following bits `01100011`, then the last (eighth) bit will map to the first coil and its value will represent True (1), the seventh bit will map to the second coil and its value will represent True (1), and so on.

4.3 C&C Channel: MQTT

We use the Message Queuing Telemetry Transport (MQTT) protocol for the implementation of the C&C control channel. MQTT is a *topic-based* PubSub protocol, designed for low-bandwidth, high-latency Machine to Machine (M2M) communication [50]. By default, it runs over TCP, it supports TLS and password-based authentication of clients. All the messages exchanged in the CPSBot botnet are addressable by topic, and their payload is data-agnostic. Topics can be hierarchically organized with different paths and each path can contain sub-paths. The set of all topics is called the topic tree, and its design is key for an effective MQTT botnet coordination.

In Figure 4 we present our implementation of a topic tree suitable to manage our CPSBot attacking a water distribution system. It includes $n + 1$ paths, where n is the number of attacked substations. The `cpsb` path contains one sub-path for each CPSBot device (in this case $n + 1$ sub-paths). For example, if a node subscribes to the `cpsb/bot1/dead` topic, then it will receive updates when the

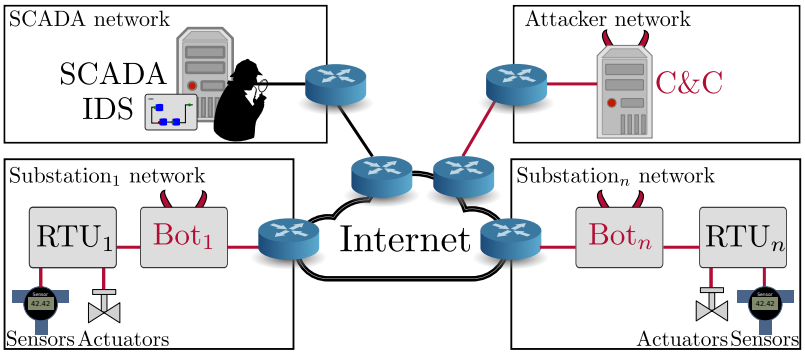


Figure 3: Centralized CPSBot attacking a water distribution ICS. There are n substations. Each substation has a remote terminal unit (RTU), and a compromised gateway device (Bot). The SCADA server is in the central control network. The attacker (via the C&C) coordinates the bots from a remote location using the red links.

bot in the first substation is disconnecting from the network (together with all the other subscribers). Another usage of this path concerns the maintenance of the bots. For example, we can use `cpsb/bot1/sw` to send binary software updates to the subscribed clients. The `subx` paths contain the information about the water distribution substations. Each sub-path manages a device (only RTUs in this case) and each sub-sub-path manages sensor and actuator values using the same memory mapping of the target RTU. For example, the `sub1/rtu1/hrs` topics contain all the messages regarding the values of the holding registers of the first RTU.

We used `mosquitto` for the MQTT broker and `paho` for the MQTT clients. Note that our MQTT setup does not depend on the target industrial protocol and with minor modifications, it can be adopted for other physical processes.

4.4 Coordination of the CPSBot Nodes

MQTT provides several useful functionalities that we are using for coordinated interactions among CPSBot bots. We comment five of them:

Message QoS. Each publish and subscribe action can be configured with a quality of service (QoS) value. There are three possible QoS values for delivering a packet at most once (QoS = 0), at least once (QoS = 1), and exactly once (QoS = 2). The default publish and subscribe QoS values are 0 and increasing QoS values result in a bigger protocol overhead. Each client subscribes to topics with a QoS and receives published messages with that QoS, even if the publisher is setting a higher QoS. We use MQTT's QoS to build a priority scheme based on the message topic. For example, we give maximum priority (QoS = 2) to messages about bot disconnections and medium priority (QoS = 1) to messages about botnet maintenance, critical sensors and actuators values, and low priority (QoS = 0) to the other messages.

Root 1	Root 2		Root n+1
cpsb/bot1/dead	sub1/rtu1/hrs/hr1	...	subn/rtu1/hrs/hr1
/bot1/...	/rtu1/hrs/...		/rtu1/hrs/...
...	...		...
/botn/...	/rtu1/cos/...		/rtu1/cos/...
/cc/...	/rtun/.../...		/rtun/.../...

Figure 4: CPSBot hierarchical topics tree design with $n + 1$ paths. The cpsb path takes care of the messages about the botnet devices. There are n sub paths, each one takes care of the messages about a water distribution substation.

Client sessions. MQTT consents to store in the broker(s) a subscription session for each subscriber (client) using a unique ID. We use this feature to optimize the clients' subscription process and message recovery. If the client session is turned on (e. g., by setting the `clean_session` flag to `False`) then the client subscribes once to the topics of interest and it does not need to resubscribe upon re-connection. Furthermore, the broker stores all the missed published messages with QoS greater than 0 and it re-publishes them when the client reconnects if the subscription was made with QoS greater than 0.

Asynchronous connections. MQTT supports both blocking (synchronous) and non-blocking (asynchronous) connections to the broker. We decided to use non-blocking clients and servers to optimize the usage of the CPSBots nodes. For example, a bot might perform other tasks while waiting to establish a connection with a broker.

Subscription with wildcards. MQTT supports meta-characters to efficiently subscribe to multiple topics. For example, to subscribe to all coils messages from the second substation a bot can subscribe to `sub2/+/cos/*`. The `+` meta-character subscribes to all the topics at the current path level, while the `*` meta-character subscribes to all the topics in the current path level and below. We use this feature in combination with our topic tree design to allow easy subscriptions to intra-substation and inter-substations topics.

Parametric keep-alive interval. MQTT is transported over TCP and sometimes one of the two hosts in a TCP stream is not working properly. This situation is called half-open connection problem and the MQTT keep-alive functionality is used to fix it. Each client can communicate the (maximum) keep-alive time in seconds which broker and client can endure without sending a message. The default keep-alive time is 60 seconds, and if it is set to 0 then this mechanism is not used. We use this feature to ensure that each communication link is working as expected and to manage the

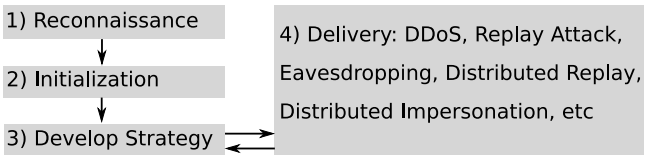


Figure 5: A CPSBot attack has four phases. Reconnaissance, Initialization, Develop Strategy, and Delivery. The first two phases are common to all CPSBot attacks. The last two are iterative.

relative geographic positions of the MQTT client and the MQTT broker. For example, clients that are geographically closer to the broker may be configured with a lower keep-alive value than the ones that are farther apart.

5 CASE STUDY: ATTACKS ON ICS

In this section, we present two cyber-physical attacks performed on simulated and real water distribution testbeds. The attacks use the centralized CPSBot implemented in Section 4. We first performed simulated attacks to speed-up the development time and to reduce the risk of damaging actual components. Then we performed the same attacks on the Water Distribution testbed (presented in Section 2.3). We start this section by describing the phases of CPSBot-based attack. Then we report on the initialization, distributed impersonation and distributed replay phases. We conclude the section with a quantitative evaluation of the presented attacks using the metrics defined in Section 3.6.

5.1 Phases of CPSBot-based Attack

A CPSBot-based attack can be decomposed into four phases based on the industrial control system cyber kill chain [26]. The attack phases are depicted in Figure 5). Firstly, we have the *Reconnaissance* phase. This is a preliminary phase where the attacker tries to get as much information as possible about the target system. Secondly, we have the *Initialization* phase. In this phase, the botmaster completes the initial infection, second infection, and rallying phases. Thirdly, we have the *Develop Strategy* phase. In this phase, the bots observe the network and the physical process and develop different attack options with the help of the C&C and the botmaster. Finally, we have the *Delivery* phase where the botmaster launches the attack(s) and tries to reach one or multiple goals. In Figure 5 we list several traditional goals such as DDoS, replay and eavesdropping and cyber-physical goals such as distributed replay and distributed impersonation. We note that CPSBot allows delivering multiple non-interfering attacks at the same time (e. g., impersonate a device while eavesdropping the communications).

In the following section, we assume that the attacker already completed the first two attack phases (that are common to all CPSBot attacks) and we discuss two advanced cyber-physical goals: *distributed impersonation* and *distributed replay*, Distributed impersonation allows the attacker to impersonate multiple gateway devices at the same time and coordinate their responses to the SCADA server. (see Section 5.3). Distributed replay enables the attacker to programmatically replay messages across substations (see Section 5.4).

5.2 Initialization Phase

Figure 6 presents an attack scenario where a CPSBot bot already infected a gateway device of WaDi. The bot, as in Figure 3, sits in between the remote terminal unit (RTU) and the access router of the substation. The bot is implementing the Infiltrator, Forger, Pub and Sub traits, indeed it is able to spoof the packets coming from its network interfaces.

Part of the initialization phase is accomplished using a combination of Ethernet bridging and forwarding techniques, see top-left ① in Figure 6. Basically, an Ethernet bridge is a virtual switch that forwards all the traffic from `eth0` to `eth1` and vice-versa. It can be configured to act as a firewall at the link and network layers. Once the bridge is established, then the bot is able to observe all the traffic between the RTU and the access router through the bridge interface (`br1`). The infected gateway devices are running Linux and we used `bridge-utils`, `iptables`, and `ebtables` tools to configure the bridges on the bots. The setup can be easily extended to bridge more than two interfaces.

Additionally, in order to optimize the CPSBot boot time, we are using the initialization phase to start the functionalities of the command and control node. Firstly, the C&C Broker is starting a `mosquitto` MQTT broker with a customizable configuration file. Then, the C&C Pub and Sub are starting asynchronous `paho` MQTT clients to establish the CPSBot network. Finally, the C&C Controller starts in idle-mode because there is no physical process data to be processed, see bottom-right ① in Figure 6.

5.3 Distributed Impersonation Phase

At the beginning of the distributed impersonation phase, the bot Infiltrator isolates the RTU from the central SCADA network. This is done by modifying the network interfaces of the bot such that `eth1` has the same IP and MAC addresses of the impersonated RTU, see center-left ② in Figure 6. Those steps are performed using a combination of `ifconfig`, `brctl` and `route` Linux commands. Then the bot starts a Modbus/TCP server listening on the same IP and port of the real RTU server. The updating period of the bot server is configurable via the time manipulation (T_M) parameter. The server is implemented using `pymodbus` and it runs in asynchronous mode using multiple threads. Once the server is running then the Pub, Sub, and Forger kick in by running the `mn.py` script. The script activates an asynchronous `paho` MQTT client that let the bot joint the CPSBot network and enables packet spoofing on the bot. Note that after these steps the bot is still connected to the RTU.

Let's assume that Figure 6 represents an impersonation attack on the first substation. Let's see how a bot is using the C&C Controller to send a valid Modbus/TCP `response` after a valid `request` from the remote SCADA, see center-right ② in Figure 6. The bot intercepts the Modbus request (Infiltrator), extracts the addresses of the requested values (Forger), and passes them to the Pub component. The Pub publishes those addresses into the relevant `sub1/rtu1` topics. The C&C Broker module, which is orchestrating the CPSBot network, collects those messages. The C&C Sub module then receives the messages containing the SCADA request addresses from the Broker and pass them to the Controller module. The Controller produces valid response values according to some estimation technique. A discussion about some potential estimation techniques

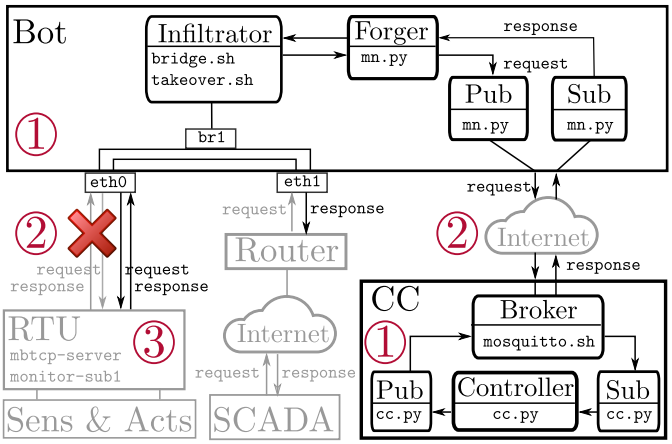


Figure 6: A CPSBot-based attack on the Water Distribution. Grey/black lines and boxes represent the ICS/CPSBot links and devices. In the initialization phase, the bot is bridging between the RTU and the router, and the C&C starts the Broker process and the Pub and Sub clients ①. In the distributed impersonation phase the bot disconnects the RTU from the network and start impersonating it by answering to the SCADA `requests` using the `responses` produced by the C&C Controller ②. The same phases take place on the other substations at the same time. Once the botmaster impersonates all the RTUs then she can target the real ones ③.

is presented in Section 6.1. The estimated response values produced by the Controller are then published by the C&C Pub module to the relevant `sub1/rtu1` topics. The bot Sub module receives the estimated responses from the C&C Broker and passes them to the Forger. Then the Forger creates a valid Modbus/TCP response packet and sends it to the SCADA via the Infiltrator.

The same impersonation technique is applied in each substation at the same time. In general, this attack is *distributed* because it disconnects all the RTUs. It is *coordinated* since each bot publishes information about its controlled substation and subscribes to information about other substations and about the status of other CPSBot devices. It is *cyber-physical* because once the botmaster is able to impersonate all the RTUs then she can target the real ones causing high-impact damages on the substations, while fooling the SCADA server, see bottom-left ③ in Figure 6.

5.4 Distributed Replay Phase

The distributed replay attack is similar to a wireless wormhole attack [12]. This type of attack enables a bot to replay locally requests and responses that are coming from other remote substations.

We now explain how to replay the content of a Modbus response from the second substation to the first one. Let's assume that the target request-response concerns the 100th holding register of the RTU in the second substation. We assume that the bots have already completed the initialization phase in the first and second substation. Then, the bot in the second substation will use a combination of `iptables` and `libnetfilter_queue` commands to extract in real time the hr100 payload from each valid RTU response. It will then publish those payloads in the `sub1/rtu1/hrs/hr100` response topic. The bot in the first substation is subscribed to all the topics concerned sub1. Indeed it will be able to reply to a hr100 SCADA request from the first RTU, with whatever hr100 value is contained in the second RTU.

We note that the same technique can be used on different device's types, on arbitrary sensor and actuator values and different substations at the same time. The result is a coordinated cyber-physical attack that can potentially alter the state of several substations without requiring detailed knowledge of the physical process by the botmaster (e. g., if attack is successful in one substation replay it on the others).

5.5 Evaluation of the CPSBot Attacks

We performed a series of measurements on the C&C and two bots while conducting the distributed impersonation and distributed replay attacks both in the simulated and real Water Distribution. For the network analysis, we used Wireshark's built-in statistics and expert information to measure delays and to flag anomalies in TCP connections. While we used Wireshark for convenience, we note that detection rules in popular IDS, such as Bro and Snort will work similarly [45], so we expect the results to be representative.

The main differences between the simulated and the real CPSBot attacks are in terms of hardware. In the simulated attack, we used MiniCPS [5] to simulate a CPSBot-based attack on Water Distribution. MiniCPS is a toolkit to perform real-time cyber-physical system simulations using lightweight virtualization and it is based on mininet [52]. For the real attacks performed on the Water Distribution testbed (see Section 2.3), we used a commercial laptop running a Linux OS to host the C&C station, and we modified Linux-based gateway devices to act as bots. The *same* code was run for the simulated and real initialization, distributed impersonation and distributed replay phases. What changed were the IP addresses (because of DHCP) and the network interface names (since they are set by the OS).

Table 2 lists the results of our evaluation using the metrics presented in Section 3.6. The SCADA polling period is in the order of few seconds (T_S) and we set the adversarial estimation period

(T_C) and the maximum traffic manipulation periods (T_M) to approximately be the half of it. Both simulated and real attacks generated one and four warning messages about TCP packets with the reset flag set. We note that those few warning messages could be avoided by improving the way the bot handles existing TCP connections after the attack starts.

Recall that μ_d denotes the average delay introduced by the CPSBot. In our experiments, μ_d was computed from the central SCADA server comparing the response times while the system was and was not under attack. Interestingly, μ_d *resulted to be close to 0 ms.* This means that our centralized CPSBot implementation did not cause significant delays in the real and simulated SCADA system. We expect that this is due to our asynchronous communications (e. g., bots do not have to periodically wait for the messages coming from the C&C) and our custom traits for the bots and the C&C (e. g., bots implementation is focused on the network manipulation while the C&C focuses on the adversarial control).

A minor issue that we experienced attacking the real testbed is related to Δ_s and Δ_r that are respectively the average time difference between the last valid RTU packet and the first packet spoofed by a bot, and the average time difference between the last SCADA request and the first valid spoofed response by a bot. We have a significant discrepancy between the attacks in the simulation framework (few milliseconds) and in the real testbed (seconds). However, this situation is experienced only one time when the attack is started, nonetheless, we are planning to conduct more experiments to better investigate it. Finally, we note that the average CPU load and memory (RAM) consumption on each bot is under 30%. This should allow performing the same attacks using devices that are even less powerful than our infected gateway devices.

6 DISCUSSION

In this section, we discuss different ways to develop an adversarial control strategy, several methods to increase the stealthiness of our CPSBots and some techniques to optimize the presented attack.

6.1 CPSBot: Control Strategies

So far, we have not discussed how the attacker should develop a suitable control strategy to take over the physical process state. We now briefly outline different potential approaches. Implementing and evaluating those approaches is out of the scope of this work, and we plan to do so in future work.

In general, the attacker has to learn how the physical process reacts to changes in actuator states, and how the state of the physical process is transitioning over time. Such knowledge can be obtained through physical process estimations [22, 31], manual data analysis, and machine learning approaches (e.g., similar to the ones employed for attack detection for CPS [40]). General learning of target CPS infrastructure has been discussed in [19].

Based on a solid understanding of the process, the attacker needs to find a sequence of actions that will lead to the goal state of the system, potentially while considering legitimate control reactions that are not under the influence of the attacker. In addition, it is likely that the physical process simulation would be optimized to remain hidden from process observers for as long as possible, or reaches its goal as soon as possible (related tradeoffs are discussed in [54]). Commonly, such control strategies require continuous tracking of the process physical evolutions using state estimation techniques such as Kalman filters and Luenberger observers.

6.2 CPSBot: Stealthiness

There are several ways to increase CPSBot stealthiness. We present a brief discussion about two strategies:

Stepping stones. We might want to introduce extra devices as intermediate proxies in the communication between the bots and the C&C [28] over the Internet. Those extra nodes might introduce secure tunnels (e. g., use Tor [44]). With this solution, we pay a penalty in terms of botnet latency and we should take into consideration if the tradeoff is worth.

C&C Protocol. We understand that using a protocol for the C&C channel that is different from the target one might increase the possibility of detection. However, this is the case also when the C&C is using the same protocol as the target system [55]. Two solutions to mitigate this problem are encryption (e. g., TLS) and obfuscation (e. g., obfuscated data structures for MQTT messages).

7 RELATED WORK

We've seen novel designs of botnets from the cyber-security field. For example, DNS botnet [4], structured and unstructured peer-to-peer botnets [42, 56], server-less botnets [58], botnet-as-a-service [11], mobile botnets [34], bitcoin-powered botnets [3] and botnets pivoting from social networks [13]. However, those designs are not addressing cyber-physical systems and OT networks.

There are several interesting analysis of traditional IT botnets. In [51], the authors managed to act as fake C&C servers and collected information about the Torpig botnet. In [2], the authors presented a system able to capture and track more than 100 unique IRC-based botnets to measure the percentage of malicious traffic attributed to those botnets on the Internet. In [35], the authors proposed a botnet take-down analysis and recommendation system. However, none of those papers analyzes a cyber-physical botnet with suitable quantitative metrics such as latency and size of the C&C packets.

There are recent academic works about cyber-physical attacks targeting several CPS devices. In particular, preferred targets are programmable logic controller (PLC). Authors discussed ransomware [20], firmware modifications [14], rootkits [1], physics-aware malware [21], and stealthy Man-in-the-Middle [53] attacks. Recently, we have seen in the wild targeted attacks on Safety Instrumented Systems (SIS) [25]. Those attacks target a single device and they are not performed using (cyber-physical) botnets.

We have seen also attempts to detect botnets for CPS. In particular, in [57] the authors are trying to detect P2P SCADA botnets by means of custom network monitoring. However, they assume to be attacked by a traditional P2P botnet.

8 CONCLUSIONS

In this work, we argue that adversarial control attacks on CPS requires a novel class of botnets that we define as *cyber-physical*. Those botnets have different requirements from traditional IT botnets such as usage of adversarial control strategies, coordinated

Table 2: Evaluation of CPSBot attacks in a simulated environment and real testbed. T_S, T_C, T_M are the SCADA, adversarial estimation and traffic manipulation periods. Δ_s is the average time difference between the last valid RTU packet and the first packet spoofed by a bot. Δ_r is the average time difference between the last SCADA request and the first valid spoofed bot response. μ_d is the additional average delay introduced by the CPSBot measured from the SCADA server. n_e is the number of Wireshark's expert info warnings and error messages. μ_{CPU} and μ_{RAM} are the approximate average CPU and RAM load on each bot. n_B is the number of controlled substations. Our optimized CPSBot implementation is able to attack the system with delay ($\mu_d = 0$) as measured at the SCADA server.

Attack	T_S [s]	T_C [s]	T_M [s]	Δ_s [ms]	Δ_r [ms]	μ_d [ms]	n_e	μ_{CPU}	μ_{RAM}	n_B
Simulated Distributed Impersonation	1.5	0.8	0.6	2.6	3.6	0.0	1	2%	20%	2
Real Distributed Impersonation	1.0	0.5	0.5	7000	7010	0.0	4	10%	30%	2
Simulated Distributed Replay	1.5	0.8	0.6	43	44	0.0	1	2%	20%	2
Real Distributed Replay	1.0	0.5	0.5	7071	7069	0.0	4	10%	30%	2

interactions among the bots, and additional constraints from the target system.

To address those challenges we presented CPSBot: a framework to build cyber-physical botnets. We leverage on a *publisher-subscriber* paradigm for the C&C channel to coordinate our bots with minimal overhead. We define an orthogonal set of *traits* to customize our bots and C&Cs according to their role in the attack and their hardware and software capabilities. For example, the bots might be specialized for packet manipulation while a C&C focuses on generating adversarial control decisions. CPSBot allows using different adversarial control strategies like machine learning classification, real-time simulation, and Kalman filtering estimation. Furthermore, we are able to adapt our botnets to different network architectures, protocols, and physical processes.

We showed the design of a centralized botnet to attack a centrally controlled CPS and a decentralized botnet to attack a CPS with distributed control. We implemented the former using MQTT for the C&C protocol and Modbus/TCP as the target network protocol. We evaluate our implementation by performing two coordinated cyber-physical attacks: *distributed eavesdropping*, and *distributed impersonation*. We evaluate our attacks with custom cyber-physical botnets metrics and we showed that our CPSBot introduces zero additional delay ($\mu_d = 0$) while the system is under attack. As result, CPSBot is able to conduct attacks that cause minimal temporal changes to the traffic, which hide the manipulation from operational alarms that might be in place.

We expect those findings on capabilities will raise awareness with stakeholders of threatened systems, and allow the defenders to design more suitable countermeasures. Potential countermeasures against our CPSBot would include close monitoring of physical process states with hardened sensors, general hardening of industrial devices against exploitation, and network segmentation and monitoring.

REFERENCES

[1] Ali Abbasi and Majid Hashemi. 2016. Ghost in the PLC: Designing an Undetectable Programmable Logic Controller Rootkit via Pin Control Attack. *Proceedings of Black Hat 2016* (2016).

[2] Moheeb Abu Rajab, Jay Zarfoss, Fabian Monrose, and Andreas Terzis. 2006. A multifaceted approach to understanding the botnet phenomenon. In *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*. ACM.

[3] Syed Taha Ali, Patrick McCorry, Peter Hyun-Jeen Lee, and Feng Hao. 2015. *ZombieCoin: Powering Next-Generation Botnets with Bitcoin.* Springer Berlin Heidelberg, Berlin, Heidelberg, 34–48. https://doi.org/10.1007/978-3-662-48051-9_3

[4] Marios Anagnostopoulos, Georgios Kambourakis, Panagiotis Kopanos, Georgios Louloudakis, and Stefanos Gritzalis. 2013. DNS amplification attack revisited. *Computers & Security* (2013).

[5] Daniele Antonioli and Nils Ole Tippenhauer. 2015. MiniCPS: A toolkit for security research on CPS Networks. In *Proceedings of the ACM Workshop on Cyber-Physical Systems-Security and/or PrivaCy (CPS-SPC)*. ACM.

[6] E. Bertino and N. Islam. 2017. Botnets and Internet of Things Security. *Computer* 50, 2 (Feb 2017), 76–79. https://doi.org/10.1109/MC.2017.62

[7] K. Birman and T. Joseph. 1987. Exploiting Virtual Synchrony in Distributed Systems. *SIGOPS Oper. Syst. Rev.* 21, 5 (Nov. 1987), 123–138. https://doi.org/10.1145/37499.37515

[8] Thomas Brandstetter. 2017. (in)Security in Building Automation: How to Create Dark Buildings with Light Speed. *Black Hat USA* 2017 (2017).

[9] Eric Byres and Justin Lowe. 2004. The myths and facts behind cyber security risks for industrial control systems. In *Proceedings of the VDE Kongress*.

[10] Antonio Carzaniga, David S Rosenblum, and Alexander L Wolf. 2001. Design and evaluation of a wide-area event notification service. *ACM Transactions on Computer Systems (TOCS)* (2001).

[11] Wentao Chang, An Wang, Aziz Mohaisen, and Songqing Chen. 2014. Characterizing botnets-as-a-service. *Proceedings of the 2014 ACM conference on SIGCOMM - SIGCOMM '14* (2014).

[12] Yih chun Hu, Adrian Perrig, and David B. Johnson. 2006. Wormhole attacks in wireless networks. *IEEE Journal on Selected Areas in Communications* (2006).

[13] Alberto Compagno, Mauro Conti, Daniele Lain, Giulio Lovisotto, and Luigi Vincenzo Mancini. 2015. Boten ELISA: A novel approach for botnet CandC in Online Social Networks. *2015 IEEE Conference on Communications and NetworkSecurity, CNS 2015* (2015).

[14] Ang Cui, Michael Costello, and Salvatore J Stolfo. 2013. When Firmware Modifications Attack: A Case Study of Embedded Exploitation.. In *Proceedings of the Network & Distributed System Security Symposium, San Diego, CA, USA*.

[15] Ang Cui and Salvatore J Stolfo. 2010. A quantitative analysis of the insecurity of embedded network devices: results of a wide-area scan. In *Proceedings of the 26th Annual Computer Security Applications Conference*. ACM, 97–106.

[16] Jörg Daubert, Mathias Fischer, Tim Grube, Stefan Schiffner, Panayotis Kikiras, and Max Mühlhäuser. 2016. AnonPubSub: Anonymous publish-subscribe overlays. *Computer Communications* (2016).

[17] Patrick Th Eugster, Pascal A Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. 2003. The many faces of publish/subscribe. *ACM computing surveys (CSUR)* (2003).

[18] Nicolas Falliere, L.O. Murchu, and Eric Chien. 2011. W32. Stuxnet Dossier. *Symantec Security Response* (2011). http://large.stanford.edu/courses/2011/ph241/grayson2/docs/w32

[19] Xuan Feng, Qiang Li, Haining Wang, and Limin Sun. 2016. Characterizing industrial control system devices on the Internet. In *Proceedings of International Conference on Network Protocols (ICNP)*. 1–10. https://doi.org/10.1109/ICNP.2016.7784407

[20] David Formby, Srikar Durbha, and Raheem Beyah. 2017. Out of Control: Ransomware for Industrial Control Systems. (2017).

[21] Luis Garcia, Ferdinand Brasser, Mehmet H Cintuglu, Ahmad-Reza Sadeghi, Osama Mohammed, and Saman A Zonouz. 2017. Hey, My Malware Knows Physics Attacking PLCs with Physical Model Aware Rootkit. In *Proceedings of the Network & Distributed System Security Symposium, San Diego, CA, USA*. 26–28.

[22] Joaquin Garcia-Alfaro, Jose Rubio-Hernan, and Luca De Cicco. 2017. On the use of watermark-based schemes to detect cyber-physical attacks. *EURASIP Journal on Information Security* 2017, 1 (2017), 8.

[23] Dieter Gollmann, Pavel Gurikov, Alexander Isakov, Marina Krotofil, Jason Larsen, and Alexander Winnicki. 2015. Cyber-physical systems security: Experimental analysis of a vinyl acetate monomer plant. In *Proceedings of the 1st ACM Workshop on Cyber-Physical System Security*. ACM.

[24] Abdulmalik Humayed, Jingqiang Lin, Fengjun Li, and Bo Luo. 2017. Cyber-Physical Systems Security–A Survey. *arXiv preprint arXiv:1701.04525* (2017).

[25] Dragos Inc. 2017. TRISIS Malware: Analysis of Safety System Targeted Malwar. https://dragos.com/blog/trisis/TRISIS-01.pdf. (2017).

[26] SANS Institute. 2015. The Industrial Control System Cyber Kill Chain. https://www.sans.org. (2015).

[27] Mihaela Ion, Giovanni Russello, and Bruno Crispo. 2010. Supporting Publication and Subscription Confidentiality in Pub/Sub Networks.. In *SecureComm*. Springer.

[28] Sheharbano Khattak, Naurin Rasheed Ramay, Kamran Riaz Khan, Affan A Syed, and Syed Ali Khayam. 2014. A taxonomy of botnet behavior, detection, and defense. *IEEE communications surveys & tutorials* (2014).

[29] Robert M Lee, Michael J Assante, and Tim Conway. 2016. Analysis of the cyber attack on the Ukrainian power grid. https://ics.sans.org/media/E-ISAC_SANS_Ukraine_DUC_5.pdf. (2016).

[30] Eireann Leverett and Frank Stajano. 2011. *Quantitatively Assessing and Visualising Industrial System Attack Surface*. Master's thesis. http://www.cryptocomb.org/2011-Leverett-industrial.pdf

[31] Lennart Ljung. 1998. System identification. In *Signal analysis and prediction*. Springer, 163–173.

[32] Matthew E Luallen. 2013. SANS SCADA and Process Control Security Survey. *A SANS Whitepaper, February* (2013).

[33] Modbus-IDA. 2006. MODBUS Application Protocol Specification V1.1b. Retrieved from: http://www.modbus.org/docs/Modbus_Application_Protocol_V1_1b.pdf. (2006).

[34] Abderrahmen Mtibaa, Khaled A. Harras, and Hussein Alnuweiri. 2015. From botnets to MobiBots: A novel malicious communication paradigm for mobile botnets. *IEEE Communications Magazine* (2015).

[35] Yacin Nadji, Manos Antonakakis, Roberto Perdisci, David Dagon, and Wenke Lee. 2013. Beheading hydras: performing effective botnet takedowns. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*. ACM.

[36] Jose Nazario. 2007. Blackenergy ddos bot analysis. *Arbor Networks* (2007).

[37] NCCIC/ICS-CERT. 2016. Industrial Control Systems Assessment Summary Report. Retrieved from: https://ics-cert.us-cert.gov/Assessments. (2016).

[38] Peter Neumann. 2007. Communication in industrial automation What is going on? *Control Engineering Practice* (2007).

[39] The Modbus Organization. 2017. Modbus FAQ: About the Protocol. Retrieved from: http://www.modbus.org/faq.php. (2017). Accessed: 2017-06-28.

[40] Mete Ozay, Inaki Esnaola, Fatos Tunay Yarman Vural, Sanjeev R Kulkarni, and H Vincent Poor. 2016. Machine learning methods for attack detection in the smart grid. *IEEE transactions on neural networks and learning systems* 27, 8 (2016), 1773–1786.

[41] Peter R Pietzuch and Jean M Bacon. 2002. Hermes: A distributed event-based middleware architecture. In *Distributed Computing Systems Workshops, 2002. Proceedings. 22nd International Conference on*. IEEE.

[42] Christian Rossow, Dennis Andriesse, Tillmann Werner, Brett Stone-Gross, Daniel Plohmann, Christian J Dietrich, and Herbert Bos. 2013. Sok: P2pwned-modeling and evaluating the resilience of peer-to-peer botnets. In *Security and Privacy (SP), 2013 IEEE Symposium on*. IEEE.

[43] Ahmad-Reza Sadeghi, Christian Wachsmann, and Michael Waidner. 2015. Security and privacy challenges in industrial internet of things. In *Design Automation Conference (DAC), 2015 52nd ACM/EDAC/IEEE*. IEEE.

[44] Amirali Sanatinia and Guevara Noubir. 2015. Onionbots: Subverting privacy infrastructure for cyber attacks. In *Dependable Systems and Networks (DSN), 2015 45th Annual IEEE/IFIP International Conference on*. IEEE, 69–80.

[45] Chris Sanders. 2017. *Practical packet analysis: Using Wireshark to solve real-world network problems*. No Starch Press.

[46] SANS. 2016. State of ICS Security Survey. Retrieved from: https://www.sans.org/reading-room/whitepapers/analyst/2016-state-ics-security-survey-37067. (2016).

[47] Nathanael Schärli, Stéphane Ducasse, Oscar Nierstrasz, and Andrew P Black. 2003. Traits: Composable units of behaviour. In *ECOOP*, Vol. 3. Springer, 248–274.

[48] Sérgio SC Silva, Rodrigo MP Silva, Raquel CG Pinto, and Ronaldo M Salles. 2013. Botnets: A survey. *Computer Networks* (2013).

[49] Jill Slay and Michael Miller. 2007. Lessons Learned from the Maroochy Water Breach. In *International Conference on Critical Infrastructure Protection*. Springer.

[50] OASIS Standard. 2014. MQTT Version 3.1.1. Retrieved from: https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html. (2014).

[51] Brett Stone-Gross, Marco Cova, Bob Gilbert, Richard Kemmerer, Christopher Kruegel, and Giovanni Vigna. 2011. Analysis of a botnet takeover. *IEEE Security & Privacy* (2011).

[52] Mininet Team. 2012. Mininet: An instant virtual network on your laptop (or other PC). (2012).

[53] David Urbina, Jairo Giraldo, Nils Ole Tippenhauer, and Alvaro Cardenas. 2016. Attacking fieldbus communications in ICS: Applications to the SWaT testbed. In *Proceedings of Singapore Cyber Security R&D Conference (SG-CRC)*.

[54] David I Urbina, Jairo Giraldo, Alvaro A Cardenas, Nils Ole Tippenhauer, Junia Valente, Mustafa Faisal, Justin Ruths, Richard Candell, and Henrik Sandberg. 2016. Limiting the Impact of Stealthy Attacks on Industrial Control Systems. In *Conference on Computer and Communications Security (CCS)*. ACM.

[55] Gernot Vormayr, Tanja Zseby, and Joachim Fabini. 2017. Botnet Communication Patterns. *IEEE Communications Surveys & Tutorials* (2017).

[56] Ping Wang, Sherri Sparks, and Cliff C Zou. 2010. An advanced hybrid peer-to-peer botnet. *IEEE Transactions on Dependable and Secure Computing* (2010).

[57] Huan Yang, Liang Cheng, and Mooi Choo Chuah. 2017. Detecting peer-to-peer botnets in SCADA systems. *2016 IEEE Globecom Workshops, GC Wkshps 2016 - Proceedings* (2017).

[58] Jianjun Zhao, Fangjiao Zhang, and Chaoge Liu. 2017. Poster : ServerLess C&C channel. (2017).