ATTACKING ELECTRIC MOTORS FOR FUN AND PROFIT

Matthew Jablonski
Radar and Radio Engineering Laboratory
George Mason University
Fairfax, VA 22030
mjablons@gmu.edu

Duminda Wijesekera
Radar and Radio Engineering Laboratory
George Mason University
Fairfax, VA 22030
dwijesek@gmu.edu

ABSTRACT

Electric motors (EMs) account for more than 40% of annual global electricity consumption and an estimated market size of $214 Billion by 2025. They drive vehicles and robots in industrial manufacturing systems, cool computers, run air-conditioners, fans and even vibrate cell phones. They are ubiquitous and controlled by hardware and software. Attacks targeting EMs can range between cyberspace and the physical world, causing damage to motors and applications that depend on them. This paper provide a comprehensive evaluation of cyber-attack objectives against EMs, which we don't believe has been done before and provide risk assessors with ways to find vulnerabilities.

We introduce the Motor Threat Model (MTM) to assist with identifying hardware and software risks in EM systems. The MTM is based on conducting a wide-scale analysis of EMs, researching different EMs and case studies of their application in real-world SCADA and transportation systems. We analyze different attack objectives against EMs based on system type and provide examples of attack techniques that can achieve the objective. Types of failures the result in these attacks include loss of control, wearing down components, limiting torque, over-rotating servo motors, fire and consequences of interfering with Pulse Width Modulation (PWM). These outcomes can be achieved by using some known attacks such as pin-control attacks disrupting PWM, denial of service (DoS) or injection network attacks, sensor attacks, and exploiting the lack of security controls of software libraries on the controller.

***K*eywords** Safety · Security · Cyber-Physical Systems · Electric Motors · Attack Model

1 Introduction

Electric motors (EMs) turn electrical energy into mechanical energy by running an electric current over a wire in a magnetic field that results in rotating a shaft that moves some payload. EMs that come in a variety of types and sizes are found in products including automated industrial platforms, transportation systems, residential applications and toys. Control systems designed to manage the operation of EM movement varies between small embedded controllers to large SCADA systems operated by humans and even robotic arms found on satellites in space operated by ground control. Creating and delivering these systems requires investments for research and development.

Historically *safety* has taken a higher priority for systems using EMs and consequent safety regulations are designed to ensure proper operating procedures, power management, and environmental conditions. Existing controls are designed to prevent accidents from unintentional issues but not intentional misuse. Thus, *cyber security risk management* in systems that leverage EMs for movement is not well defined and varies depending on the application and the perceived resultant risks due to intentional harm. This paper addresses intentional disruption of movement and control of the physical world from systems that leverage EMs, critical or otherwise, and the security controls required to ensure proper EM movement in a system that is aware of adversaries. In this paper, we address the following questions.

- *What if an attacker could inject discrete motion commands to a moving system?*,
- *What kind of attack makes the system move either unintentionally or in some manner that it was never intended?*,

- *What access is needed in order to disrupt or control movement?*

Much time and effort has been devoted to creating *modern digital* controllers use discrete instructions from hardware and software to change and control continuous motions of EMs. This could exploited to change properties such as speed or torque at different layers of a motor system (to be described shortly), although such exploits depend on the design and safety of controllers.

In this paper, we examine controllable properties and abstract away the system design to focus on deriving threats to movement applications. Using this approach, we compiled a simple *Motor Threat Model (MTM)* applicable to any EM system where the properties of the movement is managed by digital controllers. The contributions of this work are:

1. Introduction of a Motor Threat Model that characterizes motors control attacks based on the layers of an EM.
2. Demonstrating and/or enumerating practical attacks at each layer and attacker objectives.
3. Differentiating between cyber and physical attacks and the accesses required.
4. Show how the threat model can be used by defenders to generate kill chains and attack graphs specific to moving systems and thereby begin to mitigate security risks.

The rest of this paper is organized as follows: Section 2 provides a background on EMs, EM systems and model safety risks. Section 3 describes our Motor Threat Model and the seven layers of this model and how this model could be applied to real-world systems. Section 4 describes example experiments that show attacks at each of the seven layers. Section 5 provides related security research for each of the model's layers, to provide further background on movement threats. Section 6 concludes the paper.

2 Background on Electric Motor Systems

This section briefly review the functionality of EMs, controlling their movement using control theory and safety models that can used for risk management against natural failures or accidents. In-depth descriptions of motors and their controls can be found in [1] and [2].

2.1 Types of Electric Motors

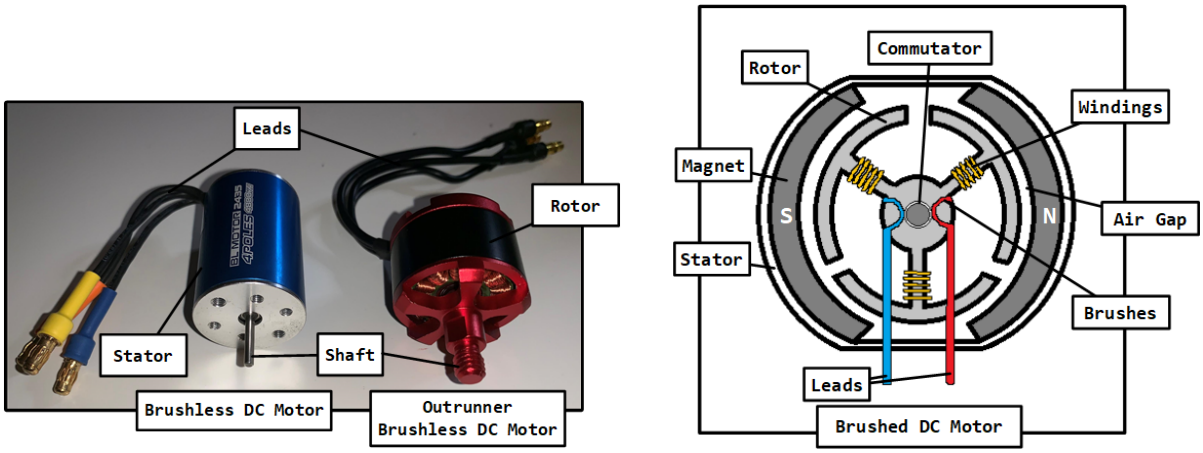
We provide a high level description of some of the main motor families, each of which contains many different subfamilies. EM families include (but are not limited to) [2]:

- **Brushed DC (BDC) motors**: Designed to run on DC power, capable of running intermittently but at very high speeds and are commonly found in toy motors, home appliances and power tools.
- **Brushless DC (BLDC) motors**: Powered by a DC electric source that uses an inverter to switch the power supply to an AC electric signal that drives the motor. They are commonly found in disk drives, printers and hobbyist air crafts.
- **Single-phase AC motors**: These motors use single-phase Alternate Current (AC), and must be started by some external force such as a auxiliary start winding or start capacitor. They are commonly found in larger home appliances, such as air conditioners, dryers or blowers.
- **Three-phase Synchronous or Asynchronous AC motors**: These motors use three-phase AC. Synchronous motors are designed so the rotor runs at the same speed of the rotating magnetic field of the stator. Asynchronous motors (AKA induction motors) have a rotating magnetic field that induces current in the rotor windings resulting in the rotor slightly lagging behind the rotating speed of the stator's magnetic field. These motors are generally found in industrial applications.

2.2 Motor Structure, Components, and Behavior

We describe the external and internal components of BDC and BLDC motors. Example externals and internals of these motors are shown in Figure 1. These components are not all-inclusive for all EM types due to space, and are described more at length in [1, 2, 3].

External Structure: The *enclosure*, or the *case*, is designed to protect the rotor, windings, and other internal components from chemical spillage, moisture, mechanical damage, and from abrasives such as dirt or water and promote air flow to ensure the motor does not become overheated during operations [4]. *Leads* are connected to power-electric drives and



(a) (b)

Figure 1: (a) Externals of inrunner and outrunner brushless motors, and (b) internals of a brushed DC motor

carry the current to and from the commutator to provide electrical power as input to the motor. The *shaft* extends from the motor's center and is connected to the output *load*, providing the mechanical energy to drive the system.

Mechanical Components: The *stator* is the stationary part of the motor that creates a magnetic field to drive the armature. The *rotor* is the component that moves as an electric current is driven through it to interact with the magnetic field. The interactions of the current-carrying conductors found on the rotor and the radial magnetic flux produced by the stator produce torque and speed. Most EMs are designed with an external stator, but some BLDC motors are structured with the rotor forming the external enclosure of the motor (known as *outrunner motors*).

BDC Internals and Behavior: The internals of the motor are designed to take input power and provide mechanical energy in the form of torque and angular speed as outputs. In BDC motors, electrical energy (current) is fed through *leads* into the *armature* (rotor) when *brushes* make contact with the *commutator*. The *windings* (or coils) are conductive wiring wrapped around the *iron cores* which provides the magnetomotive force (mmf) used to set up the magnetic flux in the *air gap*. Depending on the design, the stator may also have windings which turns the magnet into an electromagnet. To drive the motor, current enters one lead, flows through the windings and egresses through the negative lead. *Magnets* internal to the stator provide the magnetic field, as the North (N) pole exerts downward force on the current that flows through the windings and the South (S) pole exerts an upward force. Changing direction of the current flowing through the leads changes the direction of the rotor. BDC motors come in a variety of types, such as permanent magnet, series-wound, shunt-wound, compound-wound and axial field motors, depending the motor's components internally organization to generate its electrical field and result in different operating characteristics.

2.3 Torque, Speed, and Power

Motors can be designed to create rotary or linear mechanical motion. We focus on rotary motors in this paper with the purpose of describing our Motor Threat Model. We focus on torque, power and speed as these are the main output properties that an adversary would like to impact intentionally. However, there are additional parameters used to refine descriptions of motor behaviors when loads are applied, which we describe through the rest of this paper from an attacker's perspective.

2.3.1 Torque and Speed

Torque: By Newton's first law, *force*, F, is a vector product of mass times acceleration. *Torque*, τ, is a force that moves an object in a circular arc around some point, O. The distance of the object is the radius of the circle, represented by radial displacement vector r. Depending on the data sheet, torque can be measured in Newton-meters (N-m) or pound-force-feet (lb-ft), and it is defined as $\tau = rF$.

Angular and Rotational Speed: To calculate how fast an object revolves around point O, angular speed, ω, and rotational speed, n, are used for measurement. Angular speed ω is the change in angle per unit of time and the rotational speed n is the measurement of revolutions per unit of time are related as $\omega = \frac{n\pi}{30}$. Typically EM datasheets show these as revolutions per minute (RPM).

Continuous Torque and Speed: The continuous torque rating, τ_r, of an EM describes the interactions of the stator revolving field and the rotor induced field to drive the load. A motor is rated to provide a smooth continuous torque in order to smoothly drive its load at a rated power, P_r, and rated angular speed, ω_r. The continuous speed rating depends on the number of motor poles, the frequency of AC, and the amount of torque and is described as $\tau_r = \frac{P_r}{\omega_r}$ [2].

Static vs. Dynamic Torque: EMs do not always operate at their continuous rated values and constantly change as motors may need to start or stop or vary their speeds depending on the application. *Static torque* is the amount of torque produced by the EM at zero speed, as some motors do provide torque when the angular acceleration/deceleration are zero. *Dynamic torque* is determined by the rate of change over time of the system's angular momentum, M_a. M_a is a product of the polar moment of inertia of the rotating system, J_p and angular speed, ω, where $M_a = J_p\omega$. The resulting relationship describing dynamic torque can be derived as $\tau = \frac{dM_a}{dt} = \frac{d(J_p\omega)}{dt} = J_p\frac{d\omega}{dt} = J_p\alpha$ where α is the angular acceleration of the rotating [2].

2.3.2 Power

The **Output Power** (P) of an EM is used to measurement of amount energy transferred and is generally measured in Watts (W) or horsepower (HP), where 1W$\approx$ 0.00134HP. Power output P_{out} is the product of torque and speed and computed as $P_{out} = \tau\omega = \frac{\tau n\pi}{30}$.

Electrical Power: Using Ohm's law, the current going through a conductor between two points is directly proportional to the voltage between those points, and resistance is a fixed linear property of the conductor. Induction (I) is measured in *amps*, current, (V) is measured in *volts* and resistance (R) is measured in *ohms*, leading to the equation $I = \frac{V}{R}$. The electrical power input to the motor P_{elec} is defined as $P_{elec} = IV$.

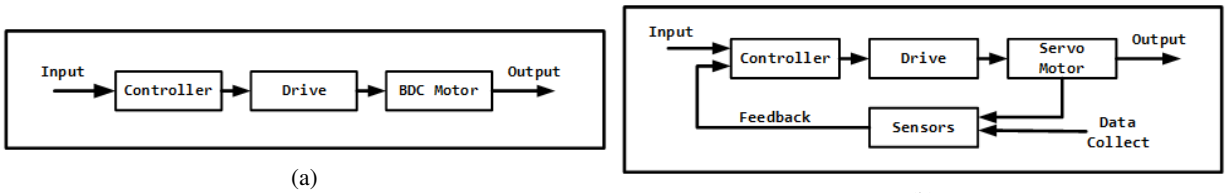
Mechanical Power: During operation, a motor draws current to spin the rotor providing torque and speed to handle the load. When there is no load, the *no-load current* I_0, is used to magnetize the iron cores of the electromagnets. The no-load current is an electrical loss, known as *iron loss*, and is subtracted from the overall current to calculate voltage loss. Using Ohm's law with armature resistance, R_a, we combine this with $P_{mech} = \tau\omega = (V - IR_a)(I - I_0)$ [1].

Efficiency: Ratio of the provided electrical energy to the produced mechanical energy is referred to as the *efficiency* of a motor, represented by η. Because of current losses in the armature, it is not possible to achieve 100% efficiency in EMs. Efficiency is defined as $\eta = \frac{P_{out}}{P_{in}} = \frac{P_{mech}}{P_{elec}} = \frac{\tau\omega}{VI} = \frac{(V-IR_m)(I-I_0)}{VI}$ [1, 5].

Back EMF: While the armature rotates, an electromagnetic (emf) force is produced as the conductor cuts through the magnetic flux following the laws of electromagnetic induction. This force results in a voltage that opposes the incoming current, and is known as *back Electro-Magnetic Force (EMF)*. The motor's output power and efficiency can be impacted by back EMF [6].

2.4 Control Theory and Motor System Design

Control theory is used to design most motor controls. Figure 2 shows both open-loop and closed-loop EM systems where the difference is that closed-loop system receives feedback from sensors outputs used by the controller to adjust the motor's operating parameters. The selection of the motor and control system components is a part of the design process and are based on work-load requirements. *Sensors* provide feedback information, such as motor temperature, speed and load size, to the controller. The *controller* makes movement decisions based on program logic and issues discrete commands to the drive controller. The *drive* controller receives the command and adjusts the current accordingly to modify speed and torque, using Pulse Width Modulation. All of these processes add latencies due to discrete communication and processing times while the motor runs continuously. *Pulse Width Modulation* (PWM) is widely used to vary the amount of current provided to the motor so that it maintains smooth movement while providing the desired output torque and speed [7], and the motor output of torque and angular speed are used to move the *load*.



(a) (b)

Figure 2: (a) Open-loop system with BDC (b) Closed-loop system with servo motor using feedback sensors [2]

2.5 Safety Regulations for Motor Systems

Historically, systems that use EMs have focused on ensuring safety from unintentional accidents. As early as 1896, the National Electrical Code added rule 8-C as the first motor safety regulation requiring *the motor and resistance box to be protected by a cutout and controlled switch*. Further safety requirements and advancements since then have added protections for items such as prolonged overloads, single phasing, short circuits, voltage issues, over-temperature, vibration, and assuring energy efficiency [8].

For the manufacturing design of EMs and generators, two standard bodies exist to ensure that the motor is designed to adhere to safety and energy efficiency requirements in harsh operational environments. In North America, the National Electric Manufacturers Association (NEMA) standards are used. NEMA MG-1 (motors and generators) defines standard classes of motors based on voltage, frequency, insulation, altitude, duty type, temperature and frame size [4]. Motor standards for the rest of the world are governed by the International Electrotechnical Commission (IEC), and these standards are comparable to the NEMA standards. Beyond standards, in the U.S., the Department of Energy regulates the energy efficiency and testing of larger electric motors in 10 C.F.R. 431, Subpart B (2019) [9] and small electric motors in 10 C.F.R. 431, Subpart X (2019) [10].

3 Threat Modeling for Motor Driven Systems

3.1 Relevance of Kill Chains to Motor Attacks

We use SANS' Industrial Control System (ICS) Cyber Kill Chain (CKC) [11] and the MITRE ICS ATT&CK Framework (currently in draft) [12] as a starting point to identify the steps that an attacker could make to control or disrupt a system's movements. Although these models are used for ICS environments, and not specific for moving systems, ICSs use EMs in a subset of components and we will show how attacks against movement have the potential to map to these frameworks. These models are defined to allow defenders to begin developing appropriate proactive countermeasures and conduct a post-compromise analysis after an attack is detected.

Based off of recent CPS malware campaign strategies, the ICS CKC model shows a successful CPS attack requires two stages with the intention of providing the defender with a strategy to detect and prevent the attack as early as possible. In Stage 1, the adversary is focused on using cyber-espionage to obtain system configuration data and persistence, and the attack indicators are like those in enterprise IT environments. In Stage 2, the adversary researches, develops, delivers, and executes some exploit that is targeted to disrupt the target physical environment [11]. Some issues in solely relying on the ICS CKC to proactively identify attacks is that insider threats, novel threats, open source research, or even reverse engineering a procured target system (such as an autonomous vehicle or parts thereof) could divulge system vulnerabilities or provide an adversary with the access needed to bypass Stage 1 altogether and execute a Stage 2 attack, especially with physical access. We describe how the Motor Threat Model allows for the refinement of the Stage 2 attack kill chain against moving systems, as we define the layers at which an attack can occur to assist with attack enumeration. By focusing on the EM, we can turn the Motor Threat Model into a multi-layered kill chain to help defenders plan accordingly for Stage 2 attacks.

The MITRE ATT&CK Framework has emerged in recent years to create a taxonomy of possible attacks in the enterprise IT environment to allow defenders to understand what attacks are being used in the wild and provide methods to attribute attacks to certain Advanced Persistent Threats (APTs) [13]. The attack steps identified in the ATT&CK Framework essentially map to Stage 1 of the CKC, as many attacks have been attributed as coming from the IT environment targeting the OT network. In 2019, MITRE has drafted the ICS ATT&CK Framework, which incorporates adversary tactics focused on *Operator Evasion*, *Disruption*, and *Destruction* that overlap with tactics for attacking movement and EMs. The ICS ATT&CK Framework is still in draft and is not publicly available at the time of this writing. In the current briefed form, it does not include tactics specifically focused on *control* of movements within a system to do something other than it was intended, as could be the case with moving systems such as robotic arms or moving vehicles. Depending on the threat model, attacks against a moving system, or even a moving component that is part of a larger system, may also be either cyber or physical. Both types of attacks may produce very similar attack outcomes when it comes to EMs. We believe that a comprehensive threat model framework should be able to identify both cyber and physical attack outcomes and will show how both can lead to similar outcomes in moving systems using the Motor Threat Model.

3.2 The Motor Threat Model

Figure 3a present the Motor Threat Model (MTM) that incorporates the interactions of an actuating motor and its digital control components. Figure 3b uses the high-level control systems design described in Figure 3a into seven

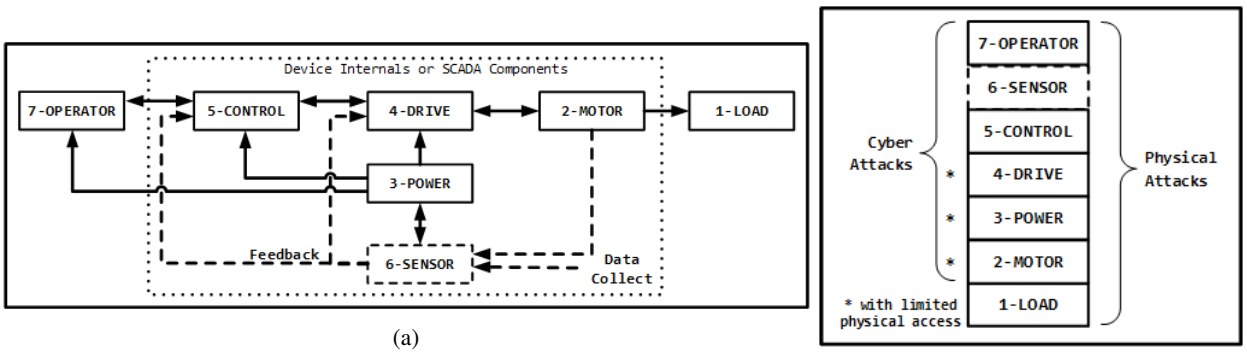


Figure 3: (a) The Motor Threat Model (b) The Seven MTM Layers and Attacks

layers in order describe different adversarial objectives and strategies. The arrows in Figure 3a represent the data, power, and movement control flow relationships between the layers. There is a dotted trust boundary surrounding the CONTROL, DRIVE, SENSOR, POWER and MOTOR layers representing either the physical enclosure protecting these components, the accessibility limitations between entities such as the internals of an electric vehicle, the internals of an industrial robot controller or the networked operational technology (OT) components of a SCADA system. Generally, the interconnections between such components are not engineered to use authentication for their interfaces with the other layers, especially as electrical current, analog sensor readings and PWM do not have authentication. Also, the SENSOR layer is optional as sensors may not be utilized in an open-loop system, as shown by the dotted interface lines. Finally, it is possible that the CONTROL and DRIVE layers could reside in a single embedded controller or a PLC, in which case a single controller could be targeted for both sets of attacks.

The 7 Layer Stack: Figure 3b shows the seven layers along with the cyber-physical attack relationships on motor driven systems where the attack objectives generally fall under three categories: (1) *Control* the system, (2) *Disrupt* the system, or (3) *Data Exfiltration* [11]. The Stage 2 attacks of the cyber-physical system (CPS) kill chains against a motor-driven system would primarily focus on controlling or disrupting the system. Assuming the adversary has obtained access using a Stage 1 attack (or bypassed Stage 1 altogether), each layer has a different attack strategy using physical to cyber techniques allowing the adversary to meet their objectives. In addition, when a lower layer is controlled or disrupted by the adversary resulting in overridden or denied service to the higher layers, the system may witness a *cascading attack* impacting the higher layers. For example, if the CONTROL layer is successfully disrupted, the DRIVE layer may still continue to function on the last controls it received and keep the system in a steady state. Component redundancies in the design and out-of-band safety monitoring systems may mitigate such attacks. We define the high-level objectives as follows:

Control: The target of control attacks depends on the adversarial capabilities, where most attacks occur using cyber or network techniques. Adversarial actions at layers 4 through 7 require knowledge about the system internals and system dynamic's precision in order to control motor movement. For example, an adversary at the highest layer can leverage OPERATOR APIs or replay commands. Adversaries with CONTROL or DRIVE layers can manipulate control, ladder logic or PWM output to move the system in ways other than it was designed, but they can also lose control over the movement without detailed system understanding.

Disrupt: Disruption can occur at any layer. Lower layer actions are more effective in disrupting actions on the motors and would be be able to circumvent higher layer actions. Attacks targeting layers 3 through 1 would only be able to disrupt or destroy the system, but not control. For example, an OPERATOR layer disruption attack may be avoided by having fail-over mechanisms, but an adversary at the CONTROL layer can override an OPERATOR's redundant controls and provide false feedback information. A LOAD layer adversary could overload and stall the motor, which may prevent movement altogether until the motor is replaced or obstruction is removed.

Data Exfiltration: In cyber attacks against moving systems, it may be the attacker's prerogative to identify important features about the system's movements, such as frequency of usage, the operational environment and failover mechanisms. This objective is also referred to as *cyber-espionage*, which is generally the focus of a Stage 1 attack in the CPS kill chain. However, the adversary may require further data during Stage 2 in order to maximize the attack impact. Just as the precision of movement controls are design requirements for the system owner, the adversary may need additional information to derive precise timing strategies to opportunistically

control or disrupt the system. Targeting of movement data may also lead to privacy concerns in some cases, especially if the data broadcasts information about personally-owned vehicles.

3.3 Attacker Modeling

In order to apply a threat model towards a system, it first helps to identify the threat actors and their layers of access. With the MTM, cyber attacks may occur at layers 5-7 over the network (CONTROL, SENSOR, OPERATOR), or at layers 2-4 with limited physical access (MOTOR, POWER, DRIVE). Physical attacks may occur at all 7 layers. Depending on the layer of the MTM, different attacker models could describe the type of attacker access, skill and resources required to pull off certain attacks against movement. We provide different attacker models below that may vary depending on the targeted moving systems, from [14].

Nation State: Attacker belonging to a state organization focused on carrying out offensive cyber campaigns. Highly skilled in both cyber and physical intrusion with extensive access to resources.

Cybercriminal: Adversary with extensive cybersecurity knowledge and skills, leverages exploits against known vulnerabilities and can possibly engineer zero-day exploits.

Terrorist: Adversary that uses computers and information technology to cause severe disruptions or widespread fear.

Hacktivist: Adversary that uses their cyber exploitation abilities to promote a political agenda.

Insider: Can include disgruntled employees or social-engineering victims, this adversary is capable of attacking a network inside of an air-gap or conducting physical attacks against otherwise inaccessible system components.

Basic User: Adversary that has established access and potentially automated attacks to target the system.

3.4 Refining Attack Objectives

With a target system and an attacker model identified, we refine the *control*, *disrupt* and *data exfiltration* objectives to describe the underlying goals and motivations for an adversary wanting to modify and manage movements in that specific system. These goals may include:

Physical Theft: By taking control of a free-moving system such as a recreational vehicle, an attacker may want to remotely bring the vehicle to them or move it somewhere that the owner cannot retrieve it. This is categorized as a *control* attack.

Altering Predictive Movement Outcomes: In systems such as 3D printers, robotic arms and surgical robots, the attacker may be interested to adjust the movement parameters resulting in variances to an otherwise predictable outcome. We categorize these as *control* or *disrupt* attacks.

Physical Damage and/or Harm: Controlling movement of heavy systems, such as movable bridges, large cranes or industrial robots in a pipeline, in a manner that may not be predictable to the owner could result in safety issues. The system could be moved a way to cause damage the system itself, the electric motor through improper usage or even the physical environment around it, resulting in loss of revenue, property, or even lives. If the proper safety controls are not in place, improper usage could result in higher current draws to the motor, resulting in the motor overheating and possibly catching fire. This is categorized as a *disruption* attack.

Halting Movement: The attacker may want to prevent usage of the system altogether, in order to stop production, halt traffic, trigger delays caused by investigations, or cause secondary damage, also is categorized as a *disruption* attack.

Intellectual Property Theft: The attacker may want to capture the movement parameters in remote systems in order to reverse engineer the system's behaviors and possibly develop competing products. IP theft could also be considered as cyber-espionage, however we describe it in this paper as focusing on reverse engineering the movements of a targeted system. This is categorized as a *data exfiltration* attack.

Privacy Invasion: By capturing sensor and other data relevant to a target system's movement, the attacker may leverage this data to track and monitor system owners. This could apply to systems such as personally owned vehicles or monitoring production on industrial assembly lines, categorized as a *data exfiltration* attack.

3.5 MTM Layers of Access

The MTM layers help visualize *where* access is required and *what* an attacker can do with that level of access. The target system and the control flow of movement instructions and physical components should be categorized according to their model layer. A complex system may have additional components at each layer, whether for redundancy or

possibly the system has many motors and moving components. We define the accesses required at each layer and high-level objectives as follows:

7 - OPERATOR: There are two levels of OPERATOR access. The first level is accessing the operator interface itself, either physically or on the network, and the second level is access to the network that serves as the OPERATOR-CONTROL communications channel. With access to the OPERATOR-CONTROL communications channel, the adversary may focus on moving laterally to the CONTROL layer. With access to either level, the adversary may *control*, *disrupt* or *exfil data* about the system's operational status, just as a normal operator would be able to do.

6 - SENSOR: There are two levels of SENSOR access. The first level is access to the sensors or wireless sensor network (WSN) and the second level is access to the out-of-band safety system, if one exists. Attacks against sensors or WSNs could be cyber or physical, with the goal being to inject data and lie about the system state or prevent communications. With access to the safety system, the adversary could either shutdown the movements in the targeted system or remove safety monitoring conditions in software to prevent safety mechanisms from shutting down the system. With access to either level, the adversary can *control*, *disrupt* or *exfil data* about the system's operational status.

5 - CONTROL: There are two levels of CONTROL access. The first level is access to the controller itself and the second level is access to the CONTROL-DRIVE communications channel (i.e. network, serial bus protocols, PWM, etc.) between the controller and its drive controller(s). With controller access in the first level, the adversary can overwrite sensor data, spoof or stop logging in memory, and issue movement commands to the drive controller(s). With only communications access, the adversary may be able to exploit protocol vulnerabilities (i.e. ICS protocols or bus protocols) to inject, spoof or jam drive control commands to achieve a similar layer of access, but they will not be able to hide the attack or prevent user feedback. With access to the CONTROL-DRIVE communications channel, the adversary could also focus on moving laterally to the DRIVE layer. With access to either level, the adversary can *control*, *disrupt* or *exfil data* about the system's operational status.

4 - DRIVE: There are two levels of DRIVE access, but both require physical access. The first level is the drive controller itself and the second level is access to the DRIVE-MOTOR channel. With controller access, the adversary could overwrite the configurations by flashing firmware or using a physically accessible management interface. Drives may be network accessible in some circumstances. With DRIVE-MOTOR channel access, the adversary is limited to attacks on electrical current fed to the EM. With access to either level, the adversary can *control* or *disrupt* movement.

3 - POWER: An adversary should be able to access the POWER layer with physical access. Power systems may have digital controls or closed control networks that could be targeted with a cyber attack. Physical access may be an insider threat scenario in restricted areas, but in moving systems this may not always be the case if the target system is parked outdoors or publicly accessible. This layer includes both external power systems or internal batteries, which depends on the system. With access to this layer, the adversary can *disrupt* movement.

2 - MOTOR: An adversary should be able to access the MOTOR layer with physical access. Digital motors, specifically digital servos, can be reprogrammed and have firmware overwritten. Physical access may be an insider threat scenario in restricted areas, but in moving systems this may not always be the case if the target system is parked outdoors or publicly accessible. With access to this layer, the adversary can *disrupt* movement.

1 - LOAD: An adversary should be able to access the LOAD layer with physical access. Physical access may be an insider threat scenario in restricted areas. This layer is the hardest layer to attack movement from in most large systems, as it may require a very large weight, lever or some other equipment to prevent movement despite the motor's attempts. With access to this layer, the adversary can *disrupt* movement.

3.6 MTM Multi-Layered Kill Chain and Attack Trees

With the threat actor, refined goals and accesses identified, we can leverage the Motor Threat Model to identify multi-layered cyber kill chains that are applicable to attacking motion control within a targeted system. Any cyber or physical attack requires first gaining access to the MTM layer. Each MTM layer has one or more levels of near-equivalent access that the attacker would need to acquire before conducting the attack. Additional steps to accomplish their goal at each layer of MTM requires some subset of the Stage 2 attack steps defined in [11]. Those steps include (1) Develop; (2) Test; (3) Deliver; (4) Install/Modify; and (5) Execute.

Understanding where accesses are required, goals of possible attack outcomes, and the steps required to achieve those goals at each of the MTM layers allows for creating potential multi-layered kill chains used to stop attacks against

movement in the system. Identified kill chains will allow defenders to exhaustively search and identify potential attack paths, or attack trees, which the attacker may take to control or disrupt movement or steal data to achieve the defined goal. The defender can then design additional security controls within the system to ensure the integrity, availability, and (when necessary) confidentiality. Figure 4 shows how these attack trees may be organized for both cyber (C) and physical (P) attacks against some target system.

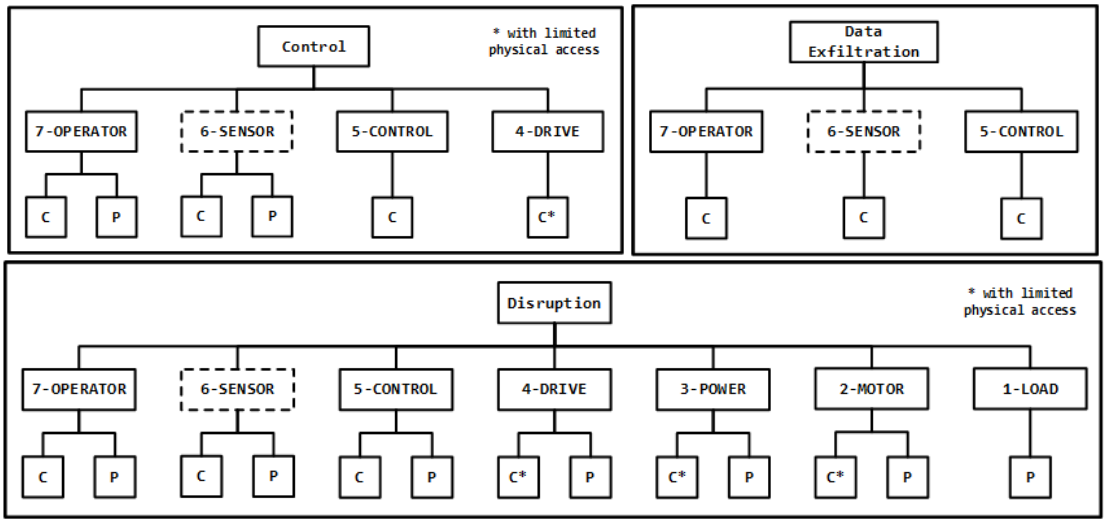


Figure 4: MTM attack tree templates for control, data exfiltration, and disrupt objectives.

4 Experimentation

This section describes experimental attacks against movement which we conducted at each layer of the Motor Threat Model (MTM). These attack experiments are included to provide examples of how movement could be targeted and attacked and the types of offensive attacks to consider when developing a threat model for a system.

4.1 OPERATOR Layer Experimentation

The OPERATOR layer attacks are application specific and has two levels of attacks: (1) attacking the operator interface; and (2) attacking the OPERATOR-CONTROL channel. We provide two example experiments against the OPERATOR-CONTROL channel.

4.1.1 OPERATOR Attack 1: Wireless Exploitation

This attack experiment demonstrate a simple wireless attack against the OPERATOR-CONTROL channel as an example of how the control channel could be overridden to control movement and/or destroy the motor by injecting commands for unintended use of an unauthenticated and used for recreational toy vehicle.

Target: A toy-grade RC vehicle shown in 5 is the attack target, where the operator controller uses one-way transmission to pair with the vehicle's receiver pair at 27.145 MHz (standard toy-grade RC communications channel) using SMD crystal oscillators. The controller has two buttons for forward and reverse movement of the toy. The receiver's controller uses a MX1608RX2 SOP16 IC to drive the on-board toy brushed DC motor to react to to the operator's commands. The vehicle is powered by 3 AA batteries for a nominal 4.5V output and new batteries were supplied at the start of testing.

Setup: The adversary uses a Dell Precision M4800 laptop, running Fedora 29 with Linux kernel version 5.1.11-200.fc29.x86_64, installed with GNU Radio 3.7.13.5, running a compatible version of GNU Radio Companion running on a HackRF software-defined radio (SDR) from Great Scott Gadgets.

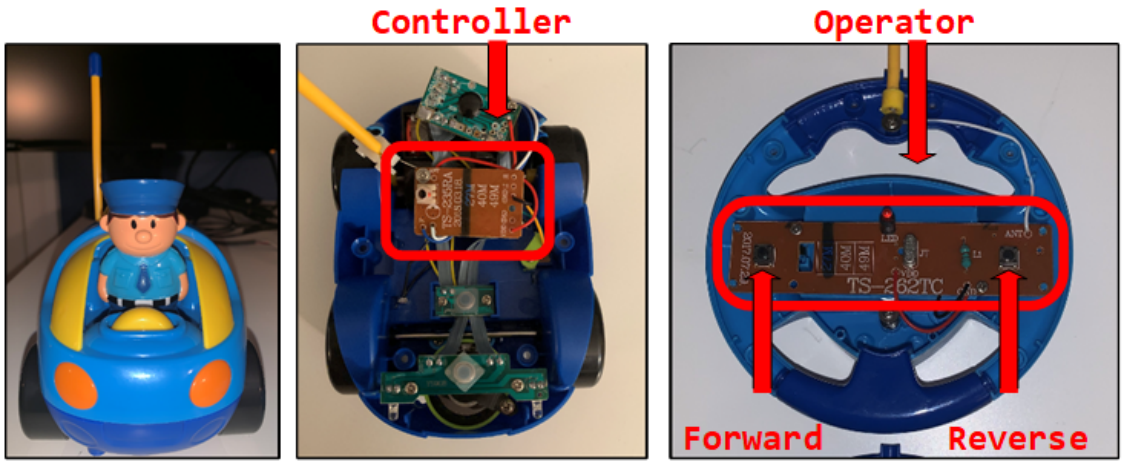


Figure 5: (Left): Target Vehicle, (Center): 27MHz embedded controller, (Right): the operator interface with two control buttons

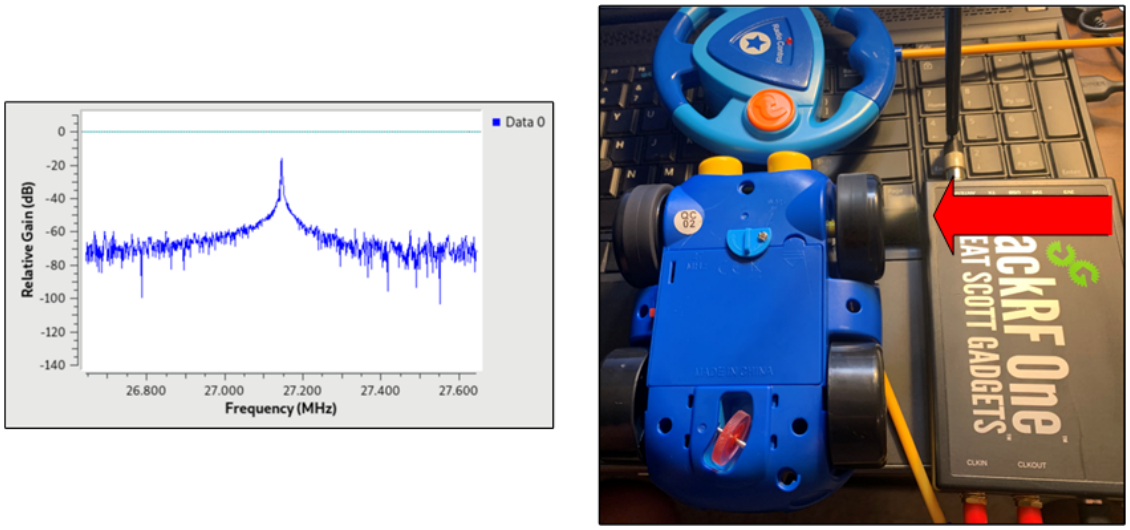


Figure 6: (Left): Capture of the 27.145MHz signal, (Right): Car's wheel rotating from replayed signal while control interface is off in background (arrow is pointing at blur which is a spinning piece of black tape)

Control: A capture of the 27.145MHZ command channel using the SDR and created two capture files, one with forward direction for 2 seconds and the other in reverse for 2 seconds. A test replay of each signal shows the car moving in either direction as desired. No decoding of the signal was necessary for this experiment. Figure 6 shows the capture signal and the moving vehicle during a replay of the signal. With a digital copy of all of the vehicle's commands, the adversary has complete control of the vehicle.

Disrupt: The files were concatenated together to produce one signal file for additional testing. The concatenated signal file was played on a loop, moving the motor forward for 2 seconds, and reverse for 2 seconds, for 15 minutes while the vehicle was turned upside down, allowing the wheels to turn freely. Measuring with a tachometer at the start, in the forward direction, the wheels spun at 690RPM forward and 681RPM reverse. After 15 minutes, both directions registered at 680RPM and the slight drop is due to voltage loss over time in the batteries.

Analysis: By capturing and splicing the movement controls, the adversary has total control over any vehicle movement (i.e. forward and backward).

4.1.2 OPERATOR Attack 2: Remote Pin Configuration Attack

This experiment is designed to demonstrate a simple command injection attack against the controller from the OPERATOR layer that is not configured securely. This is meant to simulate a Stage 2 attack.

Pigpio is a library built into the full version of Raspbian for pin control [15]. If remote GPIO is enabled, by default pigpiod listens to TCP port 8888 *as root* on all interfaces for incoming connections. The pigpio library interfaces directly with the Raspberry Pi's Broadcom SoC to control GPIO pins.

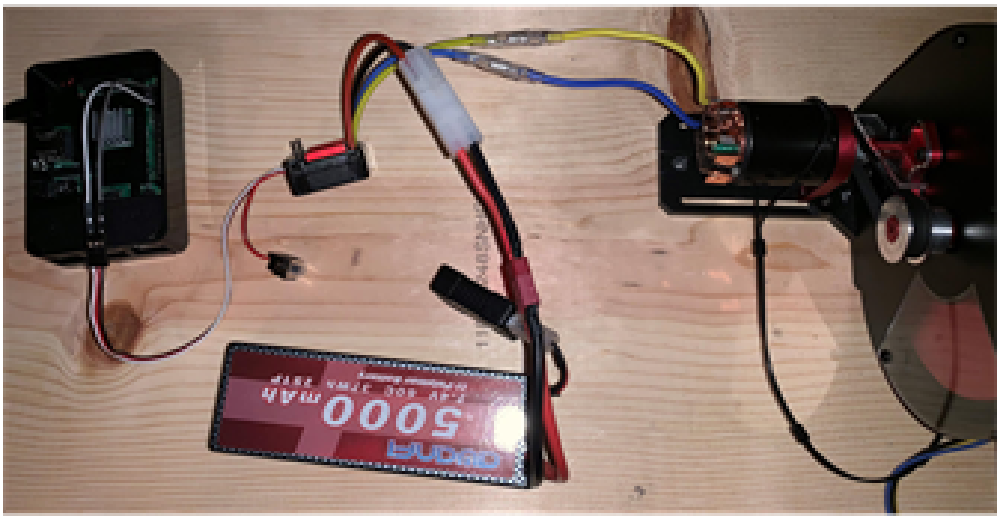


Figure 7: Target Pi connected to ESC controlling a BDC motor attached to a dynanometer, powered by a LiPO battery

Target: A sample system shown in Figure 7 was set up using a Raspberry Pi 3 B+ as a controller (Raspbian GNU/Linux 9 (stretch), Linux v. raspberrypi 4.14.71-v7+ #1145) with IP address 192.168.1.4. The Pi's GPIO18 pin is connected to a Quicrun HobbyWing 1060 Electronic Speed Controller (ESC) used to drive a brushed DC motor connected to a Minipro Electric Motor Dyno (dynanometer). The Pi uses the pigpio daemon (pigpiod v. 64) built into Raspbian, with *remote GPIO* enabled. Hardware PWM is used over GPIO18 to instruct the ESC to control the speed of the motor. The BDC motor is a generic 540 17T brushed motor, with external brushes. The dyno is used for demonstrating the impact of an attack on motor control.

Setup: The adversary uses a Dell Precision M4800 laptop, running Fedora 29 with Linux kernel version 5.1.11-200.fc29.x86_64, with the pigpio Python module and nmap v7.70 installed.

Attacker Reconnaissance: An nmap service scan is run by the adversary on the network, which identifies the target Pi as the SSH (TCP 22) version contains a string reading *Raspbian*. TCP 8888 is also open with a fingerprint string reading *NCP: DmdT*, which from our testing appears to be indicative of the listening remote GPIO pigpiod service. The adversary then use this information to blindly inject commands to GPIO pins from the network. The left image in Figure 8 shows this output and highlights the strings used for attack planning.

```
22/tcp   open  ssh            syn-ack (protocol 2.0)
| fingerprint-strings:
|   NULL:
|_    SSH-2.0-OpenSSH_7.4p1 Raspbian-10+deb9u6
| ssh-hostkey:
|   2048 7e:87:cd:3e:a5:15:70:1a:e4:80:53:01:11:01:b1:da (RSA)
| ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDL6ll97ayXbg2N1+AWcH689TS5JDzuMj1w
X7MzmP48c1YUePU4pPc7rAuxyft4O9AO3on7E7XJ/RcBtos+EZCmTTeKtKs4+AuJ04dzDkG7iX
|   256 eb:ed:c8:df:84:36:5e:f7:a2:0b:22:f6:9b:40:97:e8 (ECDSA)
| ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABB
|   256 8c:bd:de:72:90:52:a6:b9:2c:0e:2b:95:56:60:e6:e8 (ED25519)
| ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAINpwynPnUeFwtmWgEWF7o0b6rfuY1tZQvgcS
8888/tcp open  sun-answerbook? syn-ack
| fingerprint-strings:
|   NCP:
|_    DmdT
2 services unrecognized despite returning data. If you know the service/ve
==============NEXT SERVICE FINGERPRINT (SUBMIT INDIVIDUALLY)==============
```

```
from gpiozero import PWMOutputDevice
from time import sleep

motor = PWMOutputDevice(18)

motor.frequency = 250
while True:
    motor.value = 0.3
    sleep(3)
    motor.value = 0.4
    sleep(3)
~
```

Figure 8: (Left): nmap results showing Raspbian SSH and TCP 8888 pigpio signature, (Right): attack.py script used to take remote control of pin.

Control / Disrupt: With the pigpio library, if the *PIGPIO_ADDR* environmental variable is set to an IP address, the GPIO commands connect to the pigpio daemon over the network and run the commands. During a test run of the brushed motor, the adversary executed the command *PIGPIO_ADDR=192.168.1.4 python3 attack.py* from the attack laptop to run the Python script found in the right image in Figure 8. The script alternates the duty cycle between 30-40%, allowing for the motor to run at a safe speed in the lab environment. The result of the executed attack can be seen in the graph in Figure 9, which compares a baseline motor run on the target computer against when the attack script is executed in the middle of a second run using the same baseline script. The rise and fall of the motor speed in the attack run shows the alternation of motor speed between 30-40%, demonstrating adversarial control.

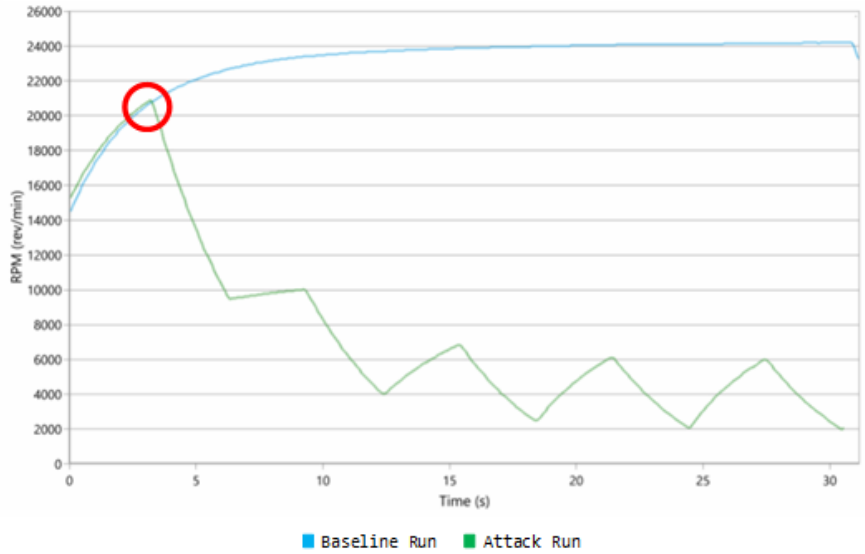


Figure 9: Results of Pigpiod Network Attack, RPM of motor vs. Time (s); The circle highlights where the attack run began, which varied the duty cycle of the motor between 30-40%.

Analysis: This attack demonstrates what an attacker could do against a target system that does not secure movement control from operators over the network, using a *remote pin configuration attack*. We targeted a specific motor application and a GPIO pin with our attack script. With blind injection, an adversary could target all GPIO pins or PWM pins, and modify their values from the network at runtime. A mitigation with this particular example may be restricting the pigpiod to only allow control from certain hosts on the local network.

4.2 SENSOR Layer Experimentation

The SENSOR layer has two levels of attacks: (1) attacking the sensor input and (2) attacking the safety system. We include a single example experiment targeting an accelerometer that is used for servo control as safety systems vary in their size and scope

4.2.1 SENSOR Attack: Accelerator Data Injection

This experiment is designed to demonstrate an adversary taking advantage of trusted input provided by a sensor. The attack shows on disrupting communications between an accelerometer and the controller communicating over an I^2C bus.

Target: A sample system was set up using a Raspberry Pi 3 B+ as a controller (Raspbian GNU/Linux 9 (stretch), Linux v. raspberrypi 4.14.71-v7+ #1145) that takes input from an ADXL345 accelerometer that provides high resolution input, using 13-bits to represent each of the x, y, and z axes. The ADXL345 communicates over the I^2C serial bus with a clock set to $\sim$70K kHZ, the target Pi uses pins GPIO2 for SDA and GPIO3 for SCL. The controller calculates the output angle for the servo using the formula, $angle = (x + y + z)\%120$, which limits the servo angle between 0-120°. The motor drive controller is a Servo Six embedded board where one servo is connected for demonstration purposes. Figure 10 shows the wiring setup using a breadboard to make the connections, the angle of the servo when the board is flat and parallel to the ground, and the angle of the servo when the board is tilted for demonstration purposes.

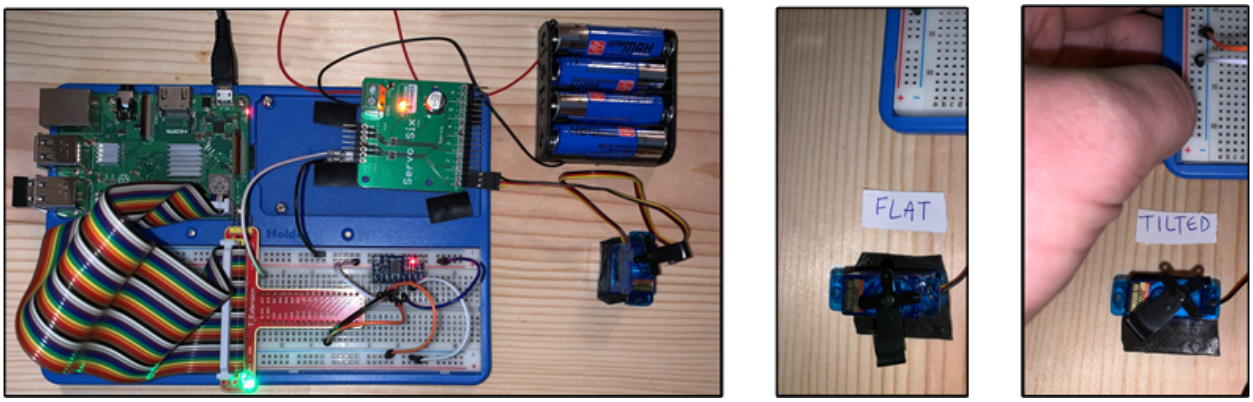


Figure 10: (Left) Target Pi setup with ADXL345 on breadboard and Servo Six servo controller, (Center) Servo angle when ADXL345 is flat, (Right) Servo angle when ADXL345 is tilted to show the difference

Setup: The adversary uses a Saleae Logic Analyzer to sniff and capture the input to devise an attack strategy. A second Raspberry Pi 3 B+ is used as an *adversarial Pi* device to simulate control and disrupt strategies.

Attacker Reconnaissance: Capturing with the Saleae shows a 6-byte response from the ADXL345 to each request. An I^{2}C capture and analysis is provided in Figure 11 showing the communications when the ADXL345 is flat on the table. The attacker Pi is configured with pins GPIO2 and GPIO3 connected in parallel with the target's connection, in order to identify the 0x53 address of the I^{2}C slave ADXL345 (the left image in Figure 12 shows this identification). With a target address and capture example, the adversarial Pi has its pin reconfigured for responding to I^{2}C requests as a slave device (RPi only permits the second I^{2}C channel for slave responses): GPIO18 is connected to SDA, GPIO19 is connected to SCL, and a ground pin is connected to the target's ground. The right image in Figure 12 shows the adversarial Pi pinout connection.

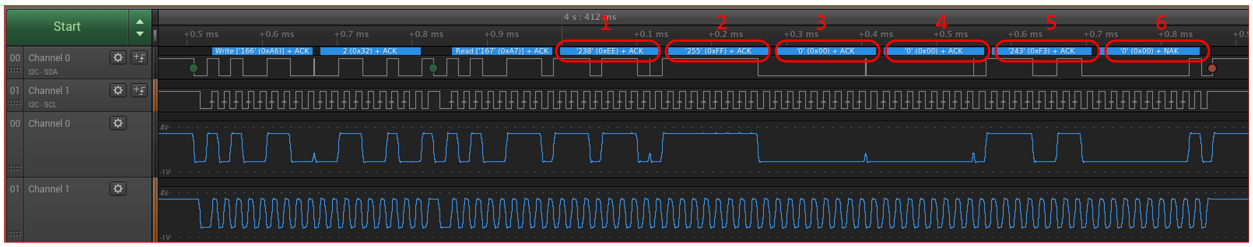


Figure 11: Logic analysis of accelerometer signal, 6 bytes are sent as a response to the I^{2}C handshake)

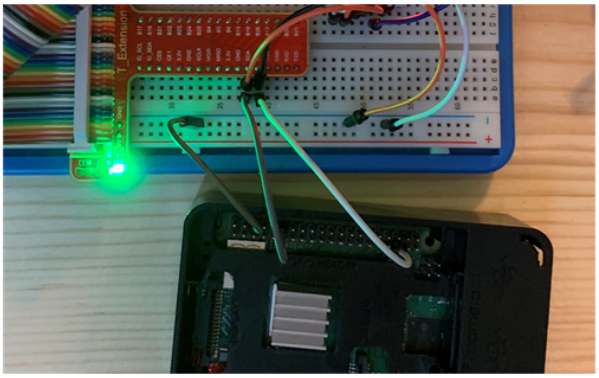


Figure 12: (Left): Address Identification of ADXL456 (0x53); (Right): Adversarial Pi Pin Configuration

Disrupt: The pigpio Python module has a function that allows for responding to I^{2}C probes, *bsc_i2c(i2c_address, data)*. The adversary sets a callback function to reply to any requests at the 0x53 address with 6 bytes of data, controlled

by the adversary. It appeared that the simultaneous response of both the adversarial Pi and accelerometer would result in a data collision, which the target Pi converts to integer values and modified the servo angle accordingly. Figure 13 shows a resulting capture of this collision using the Saleae, and the left image labeled *JAM* in Figure 14 shows the resulting servo movement.

Figure 13: Logic analysis showing colliding messages across the 6 bytes.

Control: The adversary can control the 6 bytes of input, sever the connection to the ADXL345 and respond to I^2C probes with adversary-controlled input. This gives control over the servo and allows the adversary to move it to any desired angle. The right image labeled *CONTROL* in Figure 14 shows an angle of 45°, which was the intended movement of the adversary.

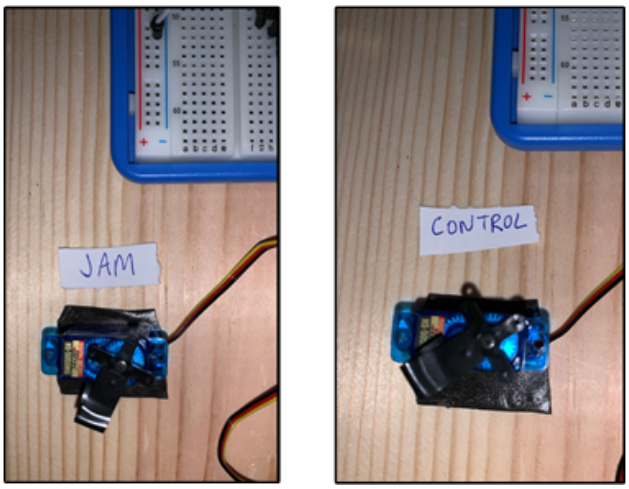


Figure 14: (Left): Servo angle resulting from the disruption attack, (Right): Servo angle resulting from the control attack.

Analysis: This attack shows two different strategies for targeting an input sensor to obtain movement control. In the disrupt scenario, the adversary did not have complete control over the 6 bytes in testing as the ADXL345's response seemed to randomly flip bits, but its possible to develop a filter to further refine this method. If the sensor was embedded in the device, the chip could be removed and a hardware implant could be utilized to inject commands over the I^2C bus without disrupting the controller. Admittedly, using a Raspberry Pi to attack a Raspberry Pi eliminates the need for mitigating any clock issues from the attacker's perspective, which is an additional challenge that may need to be overcome in real-world scenarios.

4.3 CONTROL Layer Experimentation

The CONTROL layer has two levels of attacks: (1) attacking the movement commands at runtime; and (2) attacking the CONTROL-DRIVE channel. We provide an experiment for each attack level.

4.3.1 CONTROL Attack 1: Timing Impacts of Discrete Command Injections on Motor Control

This attack is designed to show the impacts of the timing of injected commands against a controller using two different examples and the approach launching for more finely tuned control attacks.

Target: This is the same configuration setup as identified in Section 4.1.2, using a Raspberry Pi 3 B+, connected to a brushed ESC, a brushed DC motor, and the dynanometer. The target ran a script that increases the duty-cycle of the BDC motor every 0.5 seconds until the motor hits ∼16K RPM, at which point it pauses for 5 seconds, then decreases the duty cycle every 0.5 seconds until the end of the baseline run.

Setup: We assume the adversary has obtained access to the Pi controller and has the equivalent access (or greater) to the running application that controls GPIO18 PWM timing on the controller. With console access to the controller, the adversary has access to any software libraries for PIN control that may be installed on the controller. The adversary also could write their own pin control attacks by reverse engineering the memory locations for submitting requests with a copy of the software or knowing the hardware makeup of the system (we describe this further in the Section 4.4.1 experiment).

Impacts of Timing on Single Injection Attack: The left graph in Figure 15 compares a baseline run against an *attack run* where two separate commands were injected. a single command is injected at 7 seconds using the wiringPi library on the Pi to set the duty-cycle to 10%. The observed impact to the run is minimal, as the controller increased the duty-cycle every 0.5 seconds during this segment of the run. However, although minimal, a slight jerk of the motor may be disruptive in some applications. A second command is injected at 17 seconds, resulting in a large drop in motor performance as the controller stopped issuing commands for 5 seconds. The next controller command brings the angular speed of the motor back up near the levels of the baseline expected speed.

Impacts of Timing on Repeated Injection Attack: As shown by the previous example, the running application that issues faster discrete commands to motor movement will prevail in controlling the motor. The right graph in Figure 15 compares the baseline run with finely controlled adversary's commands. The adversary begins repeatedly setting the duty-cycle to 50% starting at 4 seconds and repeating every 100ms for a total of 20 seconds before stopping the attack run and relinquishing control of the motor back to the controller.

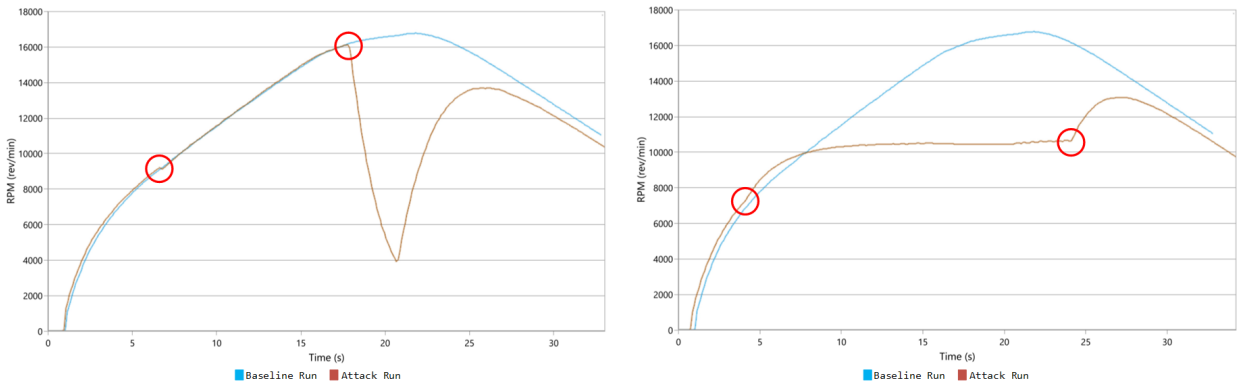


Figure 15: (Left): Single Command Injection, RPM of motor vs. Time(S) - The circles highlight a slight decrease in speed in the first command and when the second command was issued, (Right): Repeated Command Injection, RPM of motor vs. Time(s) - The circles highlight the beginning and ending of the attack run.

Analysis: Timed command injections can have varying results during an attack. If an adversary can inject discrete commands, a continuously moving motor can result in a race condition with physical world impacts. If the controller minimizes the timing between its discrete instructions to frequently instruct the speed of the motor, then a single command from an adversary may quickly jerk the motor but the impact is fixed after the controller overwrites the attack. However, if commands are sparsely issued, then a single command from an adversary can drastically change the output motor behavior. Similarly, if an adversary can issue its commands at a faster rate than the controller, then they have complete control over the system's movements.

4.3.2 CONTROL Attack 2: Hardware Implant Targeting PWM Channel

This experiment shows a control injection attack on PWM using the CONTROL-DRIVE channel by introducing a hardware implant to disrupt the control signal and take over motor control. PWM is not authenticated and a pulsating control signal caused by a hardware switch that fluctuates the voltage, usually between 0 and 3.3V or 5V. The ESC and motor receive current from an external power source, so the controller does not provide any current through this channel. This is a typical setup used to control applications that use ESCs to drive the motor, such as recreational vehicles, drones, and small personal electric vehicles such as electric skateboards.

In many motor applications, a 3-wire control channel is used to provide instructions to the drive controller from the receiver or embedded controller. This wire does not always used to provide live power and may also be used for control, such as with ESCs. Usually, the three wires are ground (black), power (red), and control (white or yellow).

Target: This is the same configuration setup as identified in Section 4.1.2, using a Raspberry Pi 3 B+, connected to a brushed ESC, a brushed DC motor, and the dynanometer. The target is running a script that increases the PWM duty-cycle of the BDC motor until the motor hits ∼24K RPM.

Setup: A second Raspberry Pi 3 B+ is used as an *adversarial Pi* device to simulate control and disrupt strategies against the PWM control channel used between the controller and ESC.

Hardware Implant Pinout: The adversarial Pi was configured with its ground pin hooked onto the black ground wire, and GPIO18 was hooked to the white control wire. Without sharing a common ground, control injection did not appear to work during testing or have any impact on motor drive. Figure 16 visually shows the relationship between the adversarial Pi and the target system, and Figure 17 shows a close up of the attached pins. A similar strategy would be utilized by an embedded hardware implant on a target application that could be physically introduced and possibly remotely controlled by the adversary.

Figure 16: Target Pi setup showing adversarial Pi acting as a hardware implant to control the CONTROL-DRIVE channel.

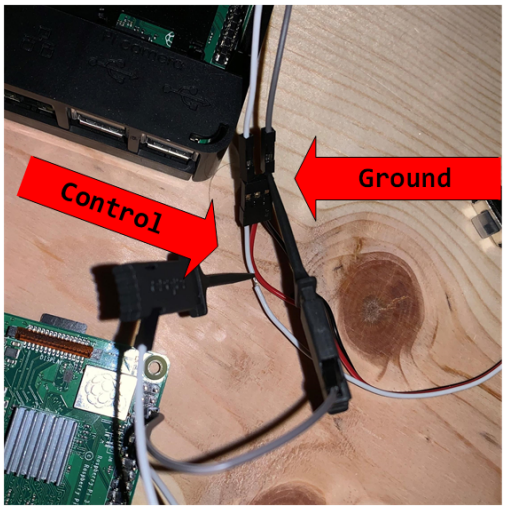


Figure 17: The adversarial Pi has its ground connected to the black wire and GPIO18 connected to the white wire to control PWM for this attack experiment

Control / Disrupt: GPIO18 was configured as an INPUT pin on the adversarial Pi until the attack started. During testing, it was observed that when the MODE of pin GPIO18 was changed to be an OUTPUT or PWM, the result would be an immediate takeover of the channel for the adversarial Pi. The target controller was leveraging wiringPi to drive the motor, with a wiringPi clock setting integer value of 100. Two separate attack runs are shown in Figure 18. The first attack run changed the status of GPIO18 to PWM mode, changed the frequency of wiringPi's clock to 150, then decreased the motor speed to ∼17K RPM, then increased the motor speed to ∼20K RPM, and concluded by changing GPIO18 back to INPUT relinquishing control back to the target's controller. The second attack run changed the status of GPIO18 to PWM mode, changed the frequency of wiringPi's clock to 50, then decreased the motor speed to ∼17K RPM and then further decreased the motor speed further to ∼16K RPM, and concluded by changing GPIO18 back to INPUT relinquishing control back to the target's controller. It appears both clock and commands were under the adversary's Pi's control in both tests.

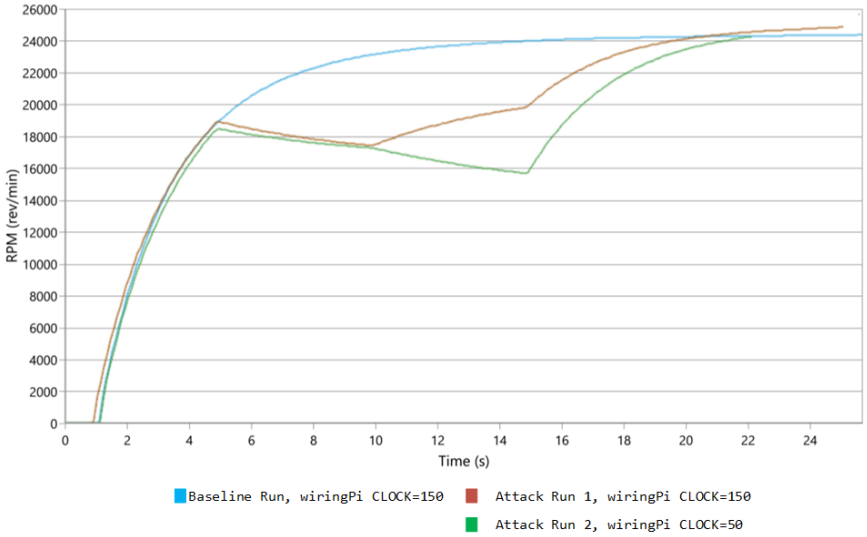


Figure 18: PWM Command Injection, RPM of motor vs. Time(S) - Attack Run 1 injected commands to slow down then slightly speed up the motor; Attack Run 2 injected commands to slow down then slightly slow down motor, again.

Analysis: If physical access is permitted, taking the approach outlined in this section to introduce a hardware implant could give an adversary an advantage. The PWM channel is not authenticated and modifications to speed settings are generally not audited and collected by a drive controller. Further, the controller seems to be blind to the attack, as its

control pin is in PWM mode. It would not be possible to use hardware interrupts to monitor bounces in the PWM pin, as PWM fluctuates voltage by design.

4.4 DRIVE Layer Experimentation

The DRIVE layer has two levels of attacks: (1) attacking the systems movements through firmware or with the ability to modify the configuration parameters, and (2) attacking the DRIVE-MOTOR channel. We provide example attack experiments for level 1, using pin control and pin configuration attacks at the drive controller. Level 2 attacks are limited to short-circuiting the connection or pulling leads, which should be detected by safety and overload protections, so an example experiment is not provided.

4.4.1 DRIVE Attack: Pin Control and Configuration Attacks

This experiment leverages a Raspberry Pi to show how an attacker could modify the pin control settings in a drive.

As noted previously in the CONTROL attack experiment in Section 4.3.1, software libraries may make assumptions during operations about the pin status at runtime, such as the pin MODE (such as INPUT, OUTPUT, PWM, etc.) or the CLOCK value. Modifications to speed, torque and position may simply opt to change the duty cycle because modifying the duty cycle can be used to smoothly change the speed and direction of a motor. We wanted to find out the effects of modifying MODE, CLOCK, and the duty cycle to an electric motor and if the behavior of the motor could change if these settings were under adversarial control.

For motor speed control (and other purposes), the Broadcom BCM2837 SoC in the Raspberry Pi 3 B+ has two PWM channels for controlling PWM output [16]. Channel PWM0 is connected to GPIO12, GPIO18, GPIO40 and GPIO52. Channel PWM1 is connected to GPIO13, GPIO19, GPIO41, GPIO45 and GPIO53. Both channels share a CLOCK *clk_pwm*, that is nominally 100MHz and can be altered by modifying its value in memory or using software libraries.

There are two modes for PWM supported by the BCM2837, MSEN0 which is the default output mode, and MSEN1 which can be used for sending data. Focusing on PWM0 and MSEN0 for driving a motor, each channel has a DATA register, *DAT1*, and a RANGE register, *RNG1*. The duty cycle is calculated per cycle as DAT1/RNG1, and is transmitted through the serial channel as part of the output signal. Similar to the CLOCK, both of these values can be modified in memory or using software libraries.

Target for DRIVE Attacks 1 and 3: This is the same configuration setup as identified in Section 4.1.2, using a Raspberry Pi 3 B+, connected to a brushed ESC, a brushed DC motor, and the dynanometer. The target is running a script that increases the duty-cycle of the BDC motor every 0.5 seconds until the motor hits ∼17K RPM, at which point it pauses for 5 seconds, then decreases the duty cycle every 0.5 seconds until the end of the baseline run.

Target for DRIVE Attacks 2: For the second attack target, we test the effects of modifying the CLOCK, DATA and RANGE and the impact on the duty cycle. We remove the dynanometer and connect pin 18 to the control input of a BLDC 45A waterproof ESC, which controls a generic BLDC motor. This setup gives more freedom and removes any risk of sparking a BDC or damaging the dyno while quickly modifying PWM duty-cycle settings.

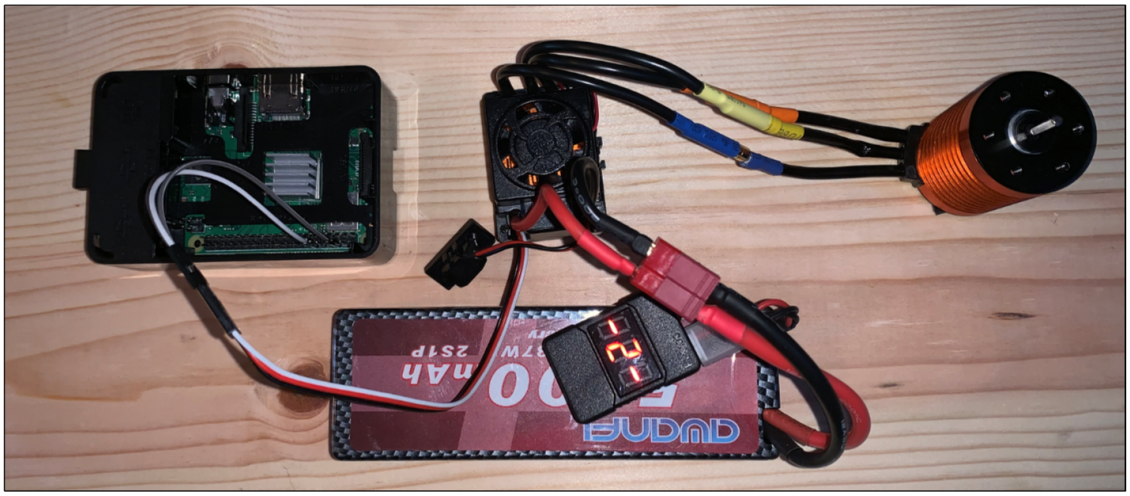


Figure 19: Target Pi setup with brushless ESC and motor target for DRIVE Attack 2.

Setup: Using the MTM definition the Pi serves as a CONTROL layer device, we designed the DRIVE layer attacks to show how reverse engineering and gaining access to the drive controller results in having complete control over the motor characteristics in memory. Drives may have limited memory for storing instructions and pin setup, depending on the drive controller model - that can range from embedded devices to full computing devices. We assume the adversary has obtained access to the Pi drive controller in this test, and has the equivalent access (or greater) to the running application that controls GPIO18 PWM timing on the controller.

4.4.1.1 Attack 1: Modify PIN Mode to INPUT

Attacker Reconnaissance: The Raspberry Pi 3 B+ uses a Broadcom BCM2837 SoC, which allows for control of GPIO pin settings in register with physical memory addresses. Reverse engineering the wiringPi library [17], which is a software library for pin manipulation of the Raspberry Pi, we mapped out the physical memory locations of CLOCK, GPIO and PWM controllers to allow the testing in this section, as Clock: 0x3F101000, GPIO: 0x3F200000 and PWM: 0x3F20C000.

By using memory mapping of these registers, we can freely modify the CLOCK, pin MODE, and PWM DATA and RANGE registers for testing. The code snippet in Figure 20, show how to memory map the CLOCK (clk), GPIO (gpio) and PWM (pwm) values. Modifying the MODE using the GPIO register, requires identifying the offset for the target pin and shifting bits in memory, depending on the target MODE. For PWM, the offsets to access DATA and RANGE integers for both PWM channels are included in the `#define` statements in Figure 20. Modifying the MODE of a pin in physical memory to INPUT, OUTPUT, PWM, etc. would take effect immediately. Changing the DATA and RANGE for each channel involves changing the value at the PWM0_DATA and PWM0_RANGE offsets within *pwm*. Modifying the clock requires stopping the clock, changing the value, and restarting.

```
#define PWM_CONTROL 0
#define PWM_STATUS  1
#define PWM0_RANGE  4
#define PWM0_DATA   5
#define PWM1_RANGE  8
#define PWM1_DATA   9
#define BLOCK_SIZE              (4*1024)

static volatile unsigned int piBase = 0x3F000000;
static volatile unsigned int *clk ;
static volatile unsigned int *gpio ;
static volatile unsigned int *pwm ;

GPIO_CLOCK_BASE = piBase + 0x00101000 ;
clk = (uint32_t *)mmap(0, BLOCK_SIZE, PROT_READ|PROT_WRITE, MAP_SHARED, fd, GPIO_CLOCK_BASE);

GPIO_BASE         = piBase + 0x00200000 ;
gpio = (uint32_t *)mmap(0, BLOCK_SIZE, PROT_READ|PROT_WRITE, MAP_SHARED, fd, GPIO_BASE);

GPIO_PWM          = piBase + 0x0020C000 ;
pwm = (uint32_t *)mmap(0, BLOCK_SIZE, PROT_READ|PROT_WRITE, MAP_SHARED, fd, GPIO_PWM);
```

Figure 20: C example code for mapping BCM2837 clk, gpio and pwm registers in memory on the RPi 3 B+.

Disrupt: A baseline run and attack run were conducted on the target Pi, as shown in Figure 21. During the attack run, the MODE of pin GPIO18 used to control the motor using hardware PWM was changed to be INPUT. As soon as the modification was made, the current was no longer supplied to the target brushed motor and the motor came to a halt.

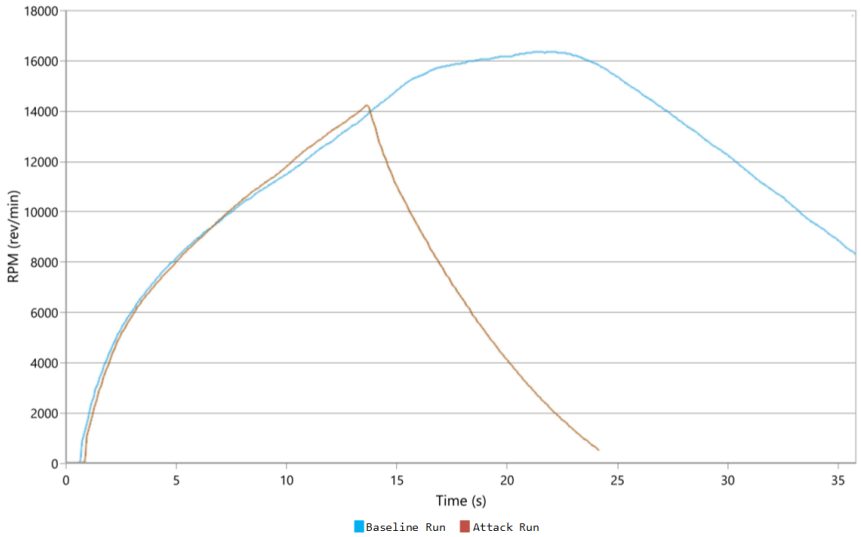


Figure 21: Results of changing pin MODE to INPUT, RPM of motor vs. Time(S)

4.4.1.2 Attack 2: Modify Clock and Duty Cycle

Attacker Reconnaissance: Broadcom's documentation notes the following, where N represents the DATA register and M represents the RANGE register [16]:

To send value N/M within a periodic sequence of M cycles, output should be 1 for N cycles and 0 for (M-N) cycles. The desired sequence should have 1's and 0's spread out as even as possible so that during any arbitrary period of time duty cycle achieves closest approximation of the value [16].

Control/Disrupt: With control over CLOCK, DATA, and RANGE registers, we wrote a C program to take an integer CLOCK value as input, modify the CLOCK to that setting and loop the DATA register from 0 to RANGE to change the duty cycle of PWM. We then recorded the values that moved the motor clockwise and counter-clockwise. We connected the BLDC ESC to GPIO pin 18 and cycled through each CLOCK and DATA value to identify the motor's behavior with respect to the clock. The results are shown in Figure 22. As the integer value for the clock increased in size, the DATA value representing the duty cycle needed to be smaller for the motor to perform at the higher clock speeds.

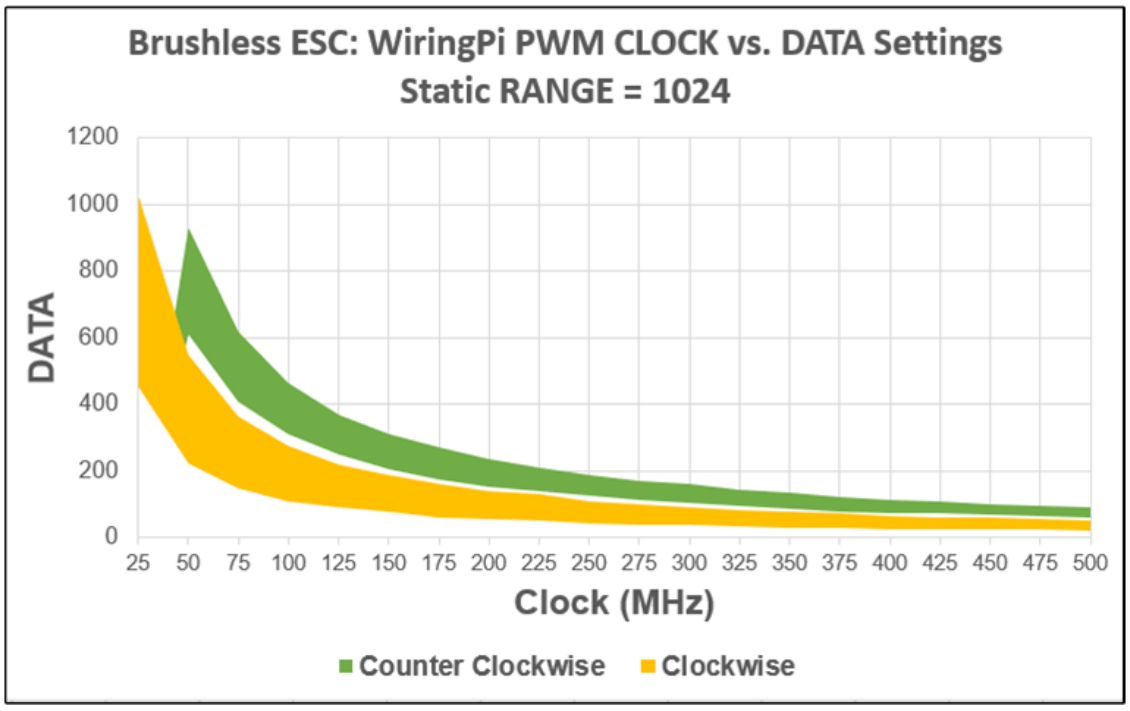


Figure 22: Results comparing movement outcomes due to CLOCK vs. DATA variations with static RANGE=1024.

4.4.1.3 Attack 3: Capturing Movement Commands in Memory

The previous attack demonstrates possibility to map the CLOCK, DATA, and RANGE values in memory of a Raspberry Pi 3 B+ in order to possibly change the functionality of a running motor. In this attack, we use those values to record movement commands and further reverse engineer movement in a target system.

Control: We created two functions to record and playback CLOCK, DATA, and RANGE values using the memory locations. The record function would run every 100ms, polling for the three values and then writing them as inputs to a file. The playback function would simply loop through the recorded value every 100ms and modify CLOCK, DATA, or RANGE if the value changed. We found as long as the recording and playback were at a rate faster than the controller made modifications, then we could playback movement commands as precisely as the controller could actuate them. The results shown in Figure 23 compare the speed of a motor in a captured baseline run against an attack run using the playbacked memory addresses, showing that the motor's angular speed curve closely matches the recorded baseline run.

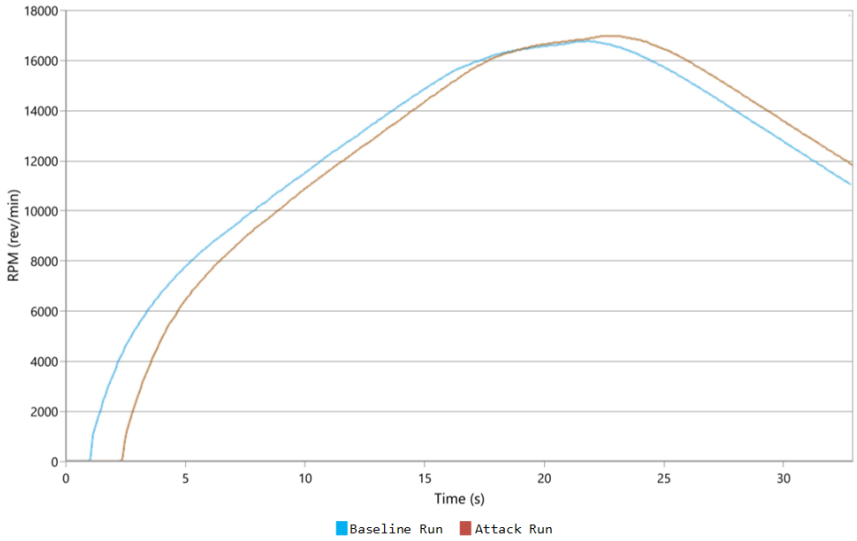


Figure 23: Results of capturing and replaying motor commands, RPM of motor vs. Time(S)

4.4.1.4 Analysis of Pin Control Attacks

The attacks outlined in this section show that root access to the drive controller may allow an adversary to control the configuration of the pin used to provide speed control to an electric motor. The results of attack 1 showed that modifications of a pin's MODE to anything other than hardware PWM can result in loss of motor function and control. The results of attack 2 showed that with root access, modifying the PWM duty cycle or CLOCK in memory can drastically change the behavior of the motor. If the target controller assumes the CLOCK is one value, but it is manipulated to another value, the motor could change direction or completely stop running. For example, if the CLOCK=100, DATA=200, and RANGE=1024 resulting in a motor running at a reasonable speed going clockwise, and the adversary modified the CLOCK value to CLOCK=175, the motor would change directions and run counter-clockwise. Similarly, it is possible to modify DATA or RANGE, however an adversary would have to constantly monitor the DATA and RANGE values and change them to their control, requiring a loop that impacts the drive performance. The results of attack 3 showed that by monitoring and capturing the memory locations of these values, it is possible to reverse engineer and playback motion controls in a target drive controller.

4.5 POWER Layer Experimentation

The POWER layer has one level of attacks typically requiring physical access to disrupt or introduce power issues. We provide an experiment to show the impact of low voltage on motors.

4.5.1 POWER Attack: Motor Performance due to Low Voltage

This experiment visualizes the impact of low voltage to an electric motor using lithium-polymer (LiPO) batteries. LiPO batteries should never be used to operate in a low voltage state. Prolonged low voltage can cause the battery to break down and possibly overheat and catch fire. LiPOs should be used with a voltage monitor that alarms if the state ever drops below the recommended voltage. Long term handling and storage of LiPO batteries is difficult, as voltage drops over time and they require regular recharging. LiPO *safes* are recommended for safe storage as they are designed to ensure any battery overheating or break down does not extend outside the safe.

Target: This experiment uses the same configuration setup identified in Section 4.1.2, using a Raspberry Pi 3 B+, connected to a brushed ESC, a brushed DC motor and the dynanometer. The target is running a script that increases the PWM duty-cycle of the BDC motor until the motor hits ∼25 RPM range.

Setup: Two separate LiPO batteries were introduced to the testing environment, one charged at 50% voltage (7.62V) and in a low-voltage state (∼6.5V), considered below the recommended alerting level of 7.4V for this particular LiPO battery. Each LiPO battery is the same model 7.4V 2 cell battery with with a capacity of ∼5000 mAh, manufactured by Awanfi INC.

Low Voltage Disruption: The baseline test run used the 7.62V LiPO and a second test run used a ∼6.5V LiPO (discharged over time due to normal usage). Prior to testing, the monitor on the ∼6.5V LiPO alerted that the voltage level was low. When the test run with the ∼6.5V LiPO began, the battery popped, almost like a gunshot, and briefly shook the test bench. The left graph in Figure 24 show that the start of the run with the ∼6.5V LiPO, the motor hesitated to increase speed (this is when the pop occurred). Then the motor speed increased to ∼16K RPM, and began to decline as the motor struggled to pull electric current to operate. By contrast, the right graph shows the starting torque between the good 7.62V LiPO (∼280 N*mm) and the low-voltage ∼6.5V LiPO (∼75N*mm), the motor torque significantly declined by about 205%. Power output also significantly declined with the ∼6.5V LiPO. As the motor began backing off from its peak speed of ∼16K RPM, the output power dropped to negative Watts, per the dynanometer.

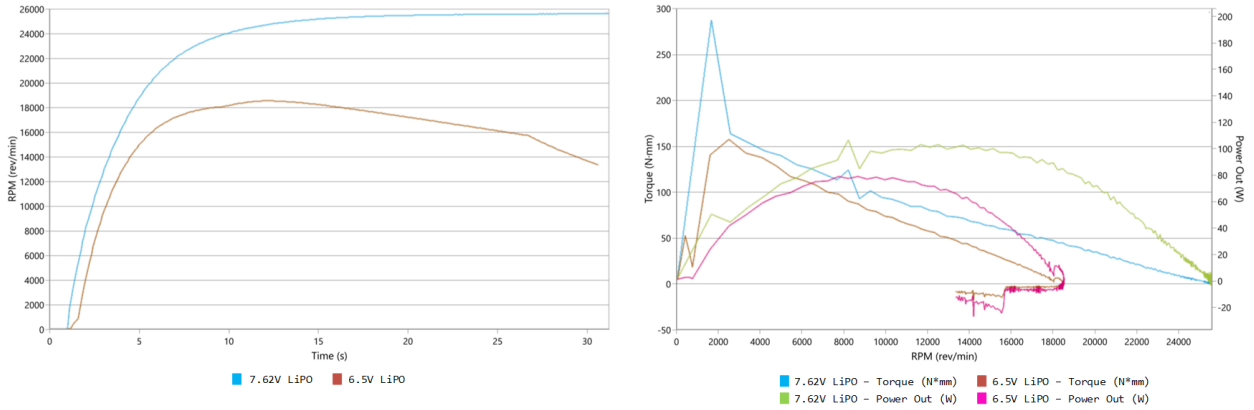


Figure 24: (Left): Comparing speed of 7.62V LiPO vs. similar low-voltage LiPO, RPM of motor vs. Time(S), (Right): Comparing torque and power output of 7.62V LiPO vs. similar low-voltage LiPO, Torque / Power Output vs. RPM of motor

Analysis: Although not an attack scenario per se, low voltage can significantly impact the performance of an electric motor, possibly causing them to stall and/or overheat as torque and power output are limited. Motors require some rated voltage in order to provide the desired performance output, and low voltage conditions result in a loss of flux density within the air gap between the rotor and stator impacting torque and output power. If an adversary tampered with power settings, resulting in a brownout scenario, the motor will under-perform and also overheat if operated for a prolonged period of time. Safety protections are generally designed into systems to limit operations when electrical power issues are a threat in operational environments.

4.6 MOTOR Layer Experimentation

The MOTOR layer has one level of attacks typically requiring physical access to the motor in order for an adversary to disrupt movements. We describe two experiments that show how adversarial physical access can change the movement characteristics of an EM.

4.6.1 MOTOR Attack 1: Motor Performance in Presence of External Electromagnet

This experiment shows the potential behavioral change of an electric motor when additional electromagnetic forces are introduced into the operational environment by introducing an electromagnet during operation.

Target: This is the same configuration setup as identified in Section 4.1.2, using a Raspberry Pi 3 B+, connected to a brushed ESC, a brushed DC motor, and the dynanometer. The target motor is a sized 540 BDC with 17T. The target is running a script that increases the PWM duty-cycle of the BDC motor until the motor hits ∼25K RPM.

Setup: An electromagnet (model WF-P50/27, produced by Wuxue Wan Fang Electric Co. Ltd.) is used for testing, which produces 500N of suction power when energized by a 12V DC power supply. The power supply and electromagnet can be seen on the test bench in Figure 25.

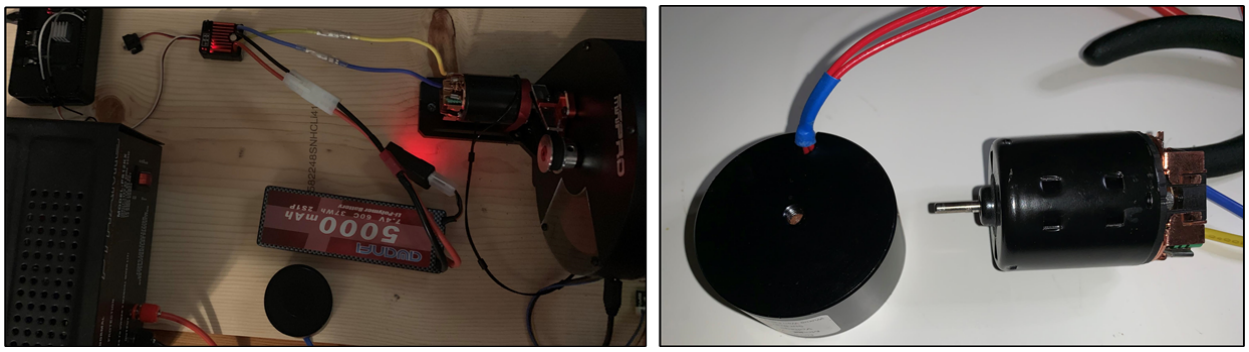


Figure 25: (Left): Test bench with electromagnet and 12V power supply; (Right): Visualization of external casing ventilation holes on BDC motor next to electromagnet.

Electromagnetic Behavioral Change: The baseline test run is demonstrated without any additional electromagnetic force. During testing, the electromagnet had no noticeable impact on the motor's performance when power was applied and the magnet was attached to the side of the motor casing without ventilation holes. When the magnet was applied over the ventilation holes, the motor's speed, torque, and power output all noticeably increased.

Figure 26 compares the results of the baseline test against two separate attack runs where the electromagnet was applied during testing. Attack run 1 had the electromagnet applied during the entire test run, resulting in a speed increase to 26K RPM or approximately 3.92% increase against the baseline run. Attack run 2 started with the applied electromagnet, and it was removed at the 15 second mark. As can be seen in the left graph in Figure 26, the removal of the electromagnet resulted in a decrease in speed back to the baseline value. The right graph in Figure 26 compares torque and power output between the baseline run and attack run 2, showing that after the electromagnet was removed, torque and power output also both returned to expected baseline values.

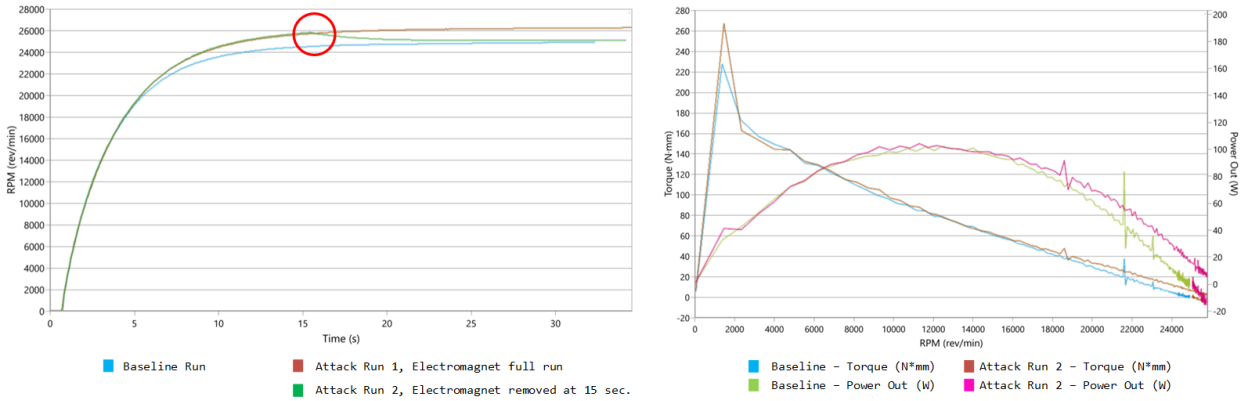


Figure 26: (Left): Electromagnet impact on speed, RPM of motor vs. Time(S); The circle highlights when the electromagnet was removed in the second attack run, (Right): Comparing torque and power output of baseline run vs. attack run 2, Torque / Power Output vs. RPM of motor

Analysis: This test was involved the intentional application of additional electromagnetic force to a test BDC on a test bench, and is not indicative of the behavior of all electric motors under similar conditions. In this experiment, the additional magnetic flux produced by the electromagnet changed the expected motor behavior by increasing speed, torque and power output. Under precisely tuned motor applications, additional electromagnetic forces may result in different behaviors other than intended by the system owner.

4.6.2 MOTOR Attack 2: Reprogramming Digital Servo Motor

This experiment shows the ease of reprogramming a digital servo motor to change its control and movement characteristics. Generally, digital servo manufacturers have programming tools available for refactoring the motor's direction, speed, failsafe, overload and deadband width properties. Many programmers have additional support to test other vendor's digital servos by providing PWM to allow for precise adjustment of the duty cycle to change the servo angle.

Target: A sample system was set up using a Raspberry Pi 3 B+ as a controller (Raspbian GNU/Linux 9 (stretch), Linux v. raspberrypi 4.14.71-v7+ #1145), leveraging the servosix library provided by the board's manufacturer for servo control [18]. The motor drive controller is a Servo Six embedded board connected to the Pi via pin GPIO17, and one Hitec HS-5055MG digital servo is connected to the servo rail for demonstration purposes. The left image in Figure 27 shows the Hitec HFP-30 digital servo programmer receiving power from the target board to make programming adjustments to the connected servo, and depicts some of the options for the digital servo. The programmer is not in the loop during test runs, and the digital servo is connected back to the servo rail on the Six Servo board for testing.

Figure 27: (Left): Servo test setup with Hitec HFP-30 programmer, (Right): Changing servo direction from CW to CCW.

Attacker Reconnaissance: Assuming the target application leverages a digital servo for movement, a programmer is procured by the adversary in order to modify the anticipated movement behaviors of the digital servo.

Disrupt: An initial series of test runs is conducted, changing the angle of the servo to 25°, 45°, 90° and 120°. The servo is disconnected from the servo rail and connected to the programmer, where the direction of the servo is changed from clockwise (CW) to counter-clockwise (CCW). The servo is reconnected to the servo rail, and the same run of tests is conducted again changing the angle of the servo to 25°, 45°, 90° and 120°. A visual comparison between the two test runs can be seen in Figures 28 and 29. The directional change reflected the angular rotation of the servo from the controller's perspective.

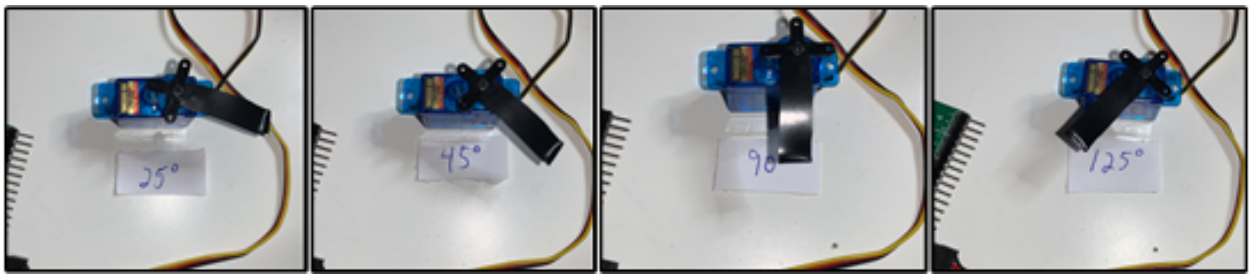


Figure 28: Directional results of initial servo run with CW movement at 25°, 45°, 90° and 120°.

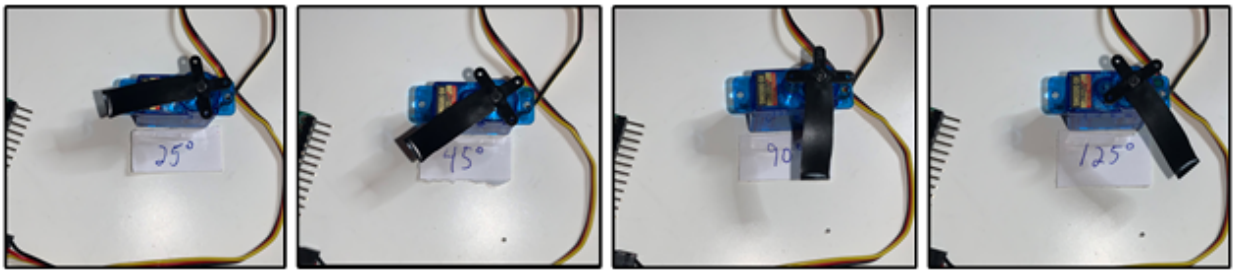


Figure 29: Reflected Results of Changing Direction of Servo Rotation from CW to CCW, showing 25°, 45°, 90° and 120°.

Analysis: By reprogramming the servo to move in a counter-clockwise direction, we changed the angular output of expected movement. Digital servos and their programmers do not leverage authentication for testing or modifying configurations. The controller may not be aware of such behavioral changes in its software unless additional sensors are leveraged monitoring the angular movement of the servo. If an intentional adjustment was made against some application, the system may be left in some unsafe state or production output may be impacted.

4.7 LOAD Layer Experimentation

The LOAD layer has one level of attacks typically requiring physical access to the applications load to disrupt movement, possibly permanently. We provide an experiment to describe the effect of stalling an electric motor under heavy load conditions.

4.7.1 LOAD Attack: Overheating and Stalling a Motor

This attack experiment is designed to show the impact of stalling and overheating an electric motor. An attack against a movement system does not necessarily need to be *cyber* in nature, as physical sabotage of the output load could be used to prohibit movement. When movement is physically prohibited, control systems are unable to issue movement commands and are limited in operational control of the system's movements.

Target and Setup: Two-speed 6" desk fans were used with the intention of stalling the motor and destroying the fans. The fans draw 0.5 Amps of current in single-phase power, and have a 2.5 Amp fuse installed for overload protection. Figure 30 shows the fan. The motor is a brushless DC motor where copper is wrapped around the base of the iron outrunner stator, energizing two windings in the stator to produce movement.

Figure 30: (Left): Target desktop fan, (Right): Obstructing movement with a pen

Disrupt: The fan was plugged into a single-phase electrical power source and turned on at the higher speed. Movement was physically obstructed by hand to monitor the temperature and ensure that a fire did not start in the test environment. Every ten minutes, the fan was allowed to run for a few seconds to measure angular speed using a tachometer. During testing, a thermometer loop was applied to measure and monitor the temperature of the stator and shaft. Figure 31 shows the graphs of speed and temperature over time. The motor burned out after 61 minutes.

During the run, the vibration of electrical energy was noticeable to the touch and audibly present. After 20 minutes, the motor had a distinct electrical burning smell that filled the test environment. When the motor finally burned out, the electrical vibration and sound came to a stop. The fuse was checked and verified that it was not blown. Figure 32 compares the dead (bad) motor and fuse against a known good motor and fuse, and there does not appear to be a visible difference with this motor.

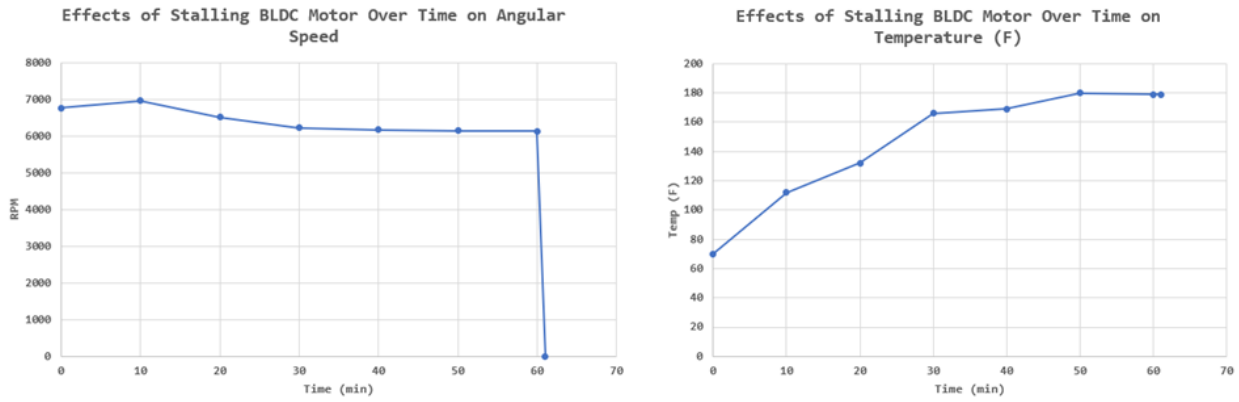


Figure 31: (Left): Comparing Speed of Stalled BLDC vs. Time (min), (Right): Comparing Temperature of the Stalled BLDC vs. Time (min)

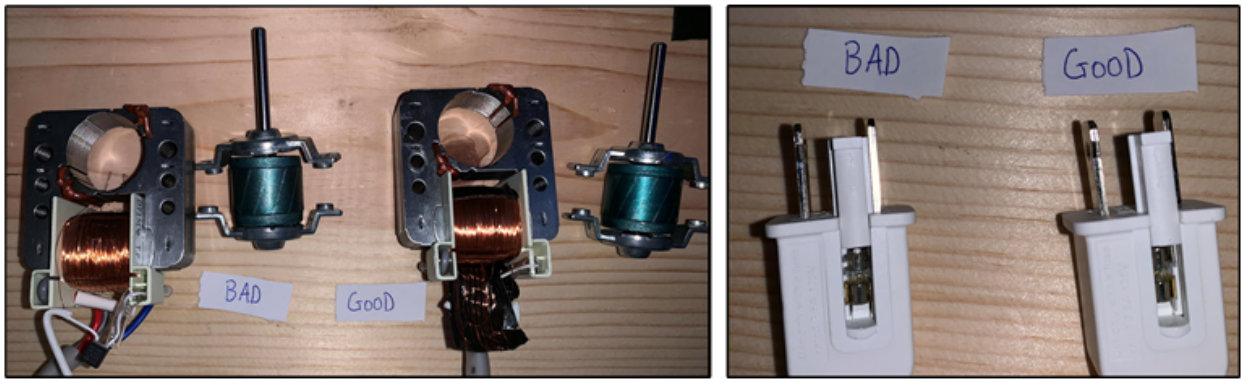


Figure 32: (Left): Visual Comparison of Dead (bad) Motor with Good Motor, (Right): Visual Comparison of the Dead (bad) Motor's Fuse with a Good Fuse; Neither comparison resulted in a noticeable visual difference

Analysis: Without protections, overloading a system could be leveraged to obstruct movement and prevent the system from moving by destroying the motor, both accidentally and intentionally. In critical operational environments, sensors and safety systems are designed to identify obstructions to movement and protect the electrical motor and moving system. Regular maintenance and motor redundancies are also used to mitigate or limit motor wear and destruction due to uneven or heavy loads.

5 Related Work

For each of the MTM layers, we can find examples in literature on identifying vulnerabilities and methods to exploit them. We provide some discussion to augment the stacked MTM model.

5.1 OPERATOR Layer Security Literature Review

Network attacks against operator interfaces in moving systems are a common issue, as confidentiality, integrity and availability can all be targeted to take control or prohibit movement. In [19], unauthenticated FTP was exploited by researchers in to perform three different attacks against an industrial robot arm from the operator interface, modifying calibration parameters, control loop parameters, and altering user inputs. In [20], an unauthenticated Bluetooth injection attack, called FacePlant, was demonstrated against an electric skateboard, where researchers showed how they reverse engineered the control protocol in order to issue a brake command to the EMs to halt the skateboard. Some motion systems are operator controlled over proprietary wireless interfaces, in which case [21] notes that such attacks could include sniffing, wardriving, replay, jamming, data link layer channel reservation abuse, evil twin, firmware update mechanisms and physical layer protocol abuse. SCADA systems are also known for their reliance on insecure protocols

for command and control [22], such as Modbus, Common Industrial Protocol (CIP) and Distributed Network Protocol (DNP3) [23].

In particular, research on attacking drones provides many examples of network attacks against the OPERATOR layer that could be applicable to other systems. The Mavlink protocol used for communicating with drones from ground stations has a number of issues, including packet forwarding attacks, eavesdropping, and hijacking [24]. Many drones provide a Wi-Fi network for command and control, allowing attacks such as de-authorising the drone owner, downloading and using the drone's app from a smart device and exploiting unauthenticated telnet, or backdoor credentials [25]. Both types of attacks were used in [26] to create HoneyDrone, which would identify attackers trying to target drones using Mavlink or Wi-Fi.

Modifications and access to operator interfaces can leave the system and surrounding humans in an unsafe state, possibly allowing the adversary to achieve some goal. In industrial robots, safety standards mandate that safety-critical operations require user confirmation. When these standards are controlled in software, [27] showed it was possible for an attacker to bypass this requirement, as they manipulated the user interface to lead the user to believe that the robot was not operational and then moved the robot while this unsafe state. In an extensive study of over 250+ HMI vulnerabilities from 2015-2016, [28] provided details regarding remote code execution, backdoor accounts, code injection, privilege escalation and admin passwords exposed through certain URLs. The BLACKENERGY2 and CRASHOVERRIDE malware campaigns from the Sandworm team, in particular, targeted libraries and configuration files of HMIs to learn more about the targeted ICS and electric grid environments [29].

5.2 SENSOR Layer Security Literature Review

Sensors have had considerable focus in recent CPS security research. We provide a background here on how the networks and physics of sensors and safety systems can be attacked, to describe some of the associated risks.

Transduction attacks can be leveraged to intentionally modify the analog readings of a sensor to manipulate a system's output or cause system errors. Malicious back-door coupling caused by resonating attacker-controlled carrier waves at the same frequency which a sensor was designed to handle for normal operation could result in such consequences to induce movement in a system [30]. For example, MicroElectro Mechanical Systems (MEMS) found in inertial sensors, such as gyroscopes or accelerometers, could be targeted using amplified acoustical waves at resonating frequencies to change and control the sensor output forcing the control system to adjust intended movements [31]. The WALNUT attack [32] demonstrated a Samsung Galaxy phone running an application that moved a toy-grade car tied to the phone's MEMS-based accelerometer could be controlled by an attacker using a controlled acoustic frequency. Similarly, the Dolphinattack showed how hidden voice commands could be introduced at inaudible frequencies received by MEMS microphones listening in AI-assistants, such as Siri, Google Now, Cortana and Alexa [33]. If smart applications that can move EM devices allow for leveraging such services, it could be possible to inaudibly control or induce movement from such an attack.

Research and development of systems monitoring using wireless sensor networks has grown over the past decade. Attacks on wireless sensor networks (WSNs), such as those used in industrial or transportation settings, include access attacks (modifying sensor data at the controller, as previously discussed), privacy attacks (capturing data to pinpoint location or system internals) and integrity attacks (jamming, spoofing, injection, etc.) [34]. Monitoring WSNs could allow for privacy concerns in moving systems, as has shown to be the case with tire pressure monitoring systems (TPMS) in automobiles [35].

Attacking the timing and positioning of moving systems using GPS sensors may also allow the adversary to control, disrupt or monitor movement. GPS was first publicly reported as a threat back in 2003, as the low power receivers found in civilian and military positioning systems are susceptible to jamming or spoofing due to the low power transmission received from satellites on the ground [36]. To this day, the U.S. Department of Homeland Security (DHS) has labeled the GPS threat the *single point of failure for critical infrastructure* [37]. Many examples of GPS attacks and mitigations can be found in literature. As an example on how to build such an attack, [38] detailed the process to develop a GPS spoofing attack using a BladeRFx40 SDR against a DJI Matrice 100 Quadcopter. As an example on how GPS spoofing could be used to fool an operator, researchers showed how an attack could introduce variations in the GPS location of passenger vehicles to result in changed directional information from smart device applications, such as Google Maps, Apple Maps, PokeMon Go, Waze, Uber, Lyft, etc [39].

As an industrial safety system example, in 2017 the TRITON malware campaign targeted a Middle Eastern oil and gas petrochemical facility's Triconex Safety Instrumented System from Schneider Electric by managing to execute a dropper file on an operator's laptop within the safety network and pivoting to the Triconex system. They reprogrammed the system, resulting in a failed state and a complete shutdown of all operational processes as a fail-safe reaction. Details reverse engineering the attack and Triconex hardware can be found in [40].

5.3 CONTROL Layer Security Literature Review

As a background, we provide some discussion here focused on what types of movement attacks are possible, given CONTROL layer access. The emerging area of robot security provides a number of examples of controller attacks and attacks against the CONTROL-DRIVE channel. In [27], researchers detailed how an adversary with controller access could modify robotic arm tuning parameters. They noted that safety and servo kinematics and configuration parameters are stored in configuration files or dynamically in memory to change the force, speed or precision of desired set-points of EMs in axis joints. In [41], a targeted attack was demonstrated on a RAVEN II tele-operated surgical robots, in which the adversary with controller access was able to stop the robot during operation or inject torque commands that caused the surgergical robot to jerk while conducting sensitive movements.

Research into PLC security has expanded over the past decade. With access to the ICS network, PLCs have been shown to be vulnerable to weak user credentials, poor access controls, or even insecure ICS network protocols which could exploited to read and write memory [42]. Such attacks to gain access may be lateral movements from the OPERATOR layer, exploiting protocols intended for operator control, or from the DRIVE layer, exploiting protocols used by the controller to communicate instructions to drives. Researchers have also found that major ICS drive controller manufacturers, Schneider-Electric, Allen-Bradley, ABB and Vacon, have designed variable frequency drives to allow unauthenticated Modbus commands to read and write motor speeds and identify critical parameters such as the critical speed of the motor [43]. As an EM-related PLC example, researchers have shown PLCs could be targeted to move EMs that lock/unlock doors in correctional facilities [44].

The Stuxnet malware campaign targeted and exploited a Seimans S7-315 in its second attack, explicitly focused on destroying centrifuge rotors by injecting commands to the centrifuge drive system. The second attack ran once a month and issued commands to the frequency drives to speed the rotors up, from 63,000 RPM to 84,600 RPM, for fifteen minutes. The second month the attack sequence ran, it was modified to drop rotor speed down to 120 RPM, only to speed them up again. It was unclear how effective these modifications were in destroying centrifuges [45]. Although centrifuge rotors operate similarly to rotors in electric motors, we note they are not the same and this is only an example of modifying speed with the intent of causing destruction.

5.4 DRIVE Layer Security Literature Review

Drives controllers have not had as much security research focus, possibly because their designs focus on availability over confidentiality and integrity. We provide some details here as to how drives can be accessed, managed, and possibly exploited.

Hobbyists have been hacking and modifying ESC firmware and hardware for years to improve performance, speed, and torque in their applications. Open source firmware, such as SimonK [46], is widely used to reflash ATmega-based microcontrollers found in commercial ESCs. As a recent open source hardware example, Electronoobs' PCB schematic can be downloaded, printed, configured and programmed using Arduino C [47].

Networked variable speed drives (VSDs) have remote management capabilities, but we did not identify publicly available research focused on vulnerability exploitation of VSDs. To provide an idea of what types of vulnerabilities may be present in networked VSDs, we will provide details regarding CVEs reported by ICS-CERT on VSDs and variable frequency drives (VFDs). A DoS exploit was recently identified in Rockwell Automation's PowerFlex 525, detailed in [48], which could allow an attacker to deny services to the Common Industrial Protocol (CIP). This particular vulnerability would result in denying new client connections to issue movement commands while allowing the attacker to continue to send movement commands (CVE-2018-19282). In 2018, a stack-based buffer overflow in Eaton 9000X drive (used in industrial manufacturing) was identified allowing for remote code execution (CVE-2018-8847) [49]. Also in 2018, researchers identified 3 CVEs in Fuji Electric's Alpha 5 Smart servo system, including a heap-based buffer overflow allowing for arbitrary code execution and a buffer overflow allowing for privilege escalation [50].

In 2016, [51] noted that an attacker with access to a controller could modify the internal pin control settings via 2 different attacks. In the first pin control attack, with root access, an attacker could change the control of a LED pin from OUTPUT to INPUT, resulting in a lack of control of the pin from the controlling application. If the pin was being used to drive PWM for a motor under such an attack, the motor could be stopped and prevented from moving, as we described in this paper. In the second pin configuration attack, with access equivalent to the running application, an attacker could modify the pin settings and still disable the pin. As control to the pin output is assumed by a drive application, unless the pin's settings are continuously monitored and checked, an attack could modify the duty cycle of the motor at the pin level as a TOCTOU attack and change the motor's drive settings. We demonstrated the effects of these attacks using experimentation earlier in the paper.

5.5 POWER Layer Security Literature Review

Attacks against the POWER layer are mostly limited to physical access attacks. However, we note that [52] showed how to modify the firmware on embedded controllers found in lithium ion batteries by targeting Smart Battery Specification (SBS) communications over the SMBus [52]. They demonstrated that by targeting a battery's controller, it is possible to change settings beyond safety specifications such as the charge level of the battery or current flow, which could result in overheating or fire.

5.6 MOTOR Layer Security Literature Review

MOTOR layer attacks are mostly physical access attacks. In 2011, the Illinois Statewide Terrorism and Intelligence Center released a report entitled *Public Water District Cyber Intrusion* detailing how Russian hackers were able to turn the system on and off resulting in the burnout of a water pump (electric water pumps use EMs to pump water). A week later, DHS and the FBI concluded that it was not a cyber-related attack. The scare was caused when the system owner logged into the SCADA system while on vacation from Russia several months before the incident. When the pump later burned out, the SCADA logs were reviewed, and the Russian IP address in the logs became the focus of the investigation [53].

5.7 LOAD Layer Security Literature Review

LOAD layer attacks are also mostly limited to physical access attacks. We were unable to find examples in security literature of disrupting movement in control by specifically targeting the load. Preventing system overload risks is typically a focus of critical systems safety engineering where typically with multiple layers of safety controls. Discussion about how to size a motor and design a system to handle loads can be found extensively in [2]. Overloading a system under adverse conditions can result in the failure of mechanical components in a motor, such as the bearings, or cause overheating leading to insulation failure of the windings.

6 Conclusion

This paper focused on addressing the challenge of securing digital systems that leverage electric motors to move and interact with the physical world. These problems can be summarized in three ways: (1) rotary motors are designed to move continuously but digital command over these movements involves discrete instructions varying the speed, torque, and power output of the system; (2) multiple layers are required for motor control, especially in complex systems, and securing all of these layers and channels is not possible especially if physical access is permitted; and (3) existing threat models do not focus specifically on threat modeling actuation and movement within these systems.

To address these challenges, we introduced the Motor Threat Model as a starting point in identifying attack vectors within moving systems that were designed following the principles outlined in control theory. The model was intended to be simple and show the relationships between both cyber and physical attacks against a system along with an attacker's goals and potential outcomes. We provided discussion on how to leverage the model to create attack trees and multi-layered kill chains to allow defenders to identify security vulnerabilities within these systems. After a reviewing multiple systems that leverage electric motors, we proposed this layered approach that fits most electric motor control systems under most conditions.

To augment our model, we provided attack experiments at each of the layers to demonstrate the related security concerns. Similar attacks and approaches may be leveraged by security researchers targeting other systems in order to identify potential physical outcomes that could be harmful to society. We also provided a security literature review intended to provide a background on attacks at each of the layers, which also showed gaps in security research, especially at the lower layers.

References

[1] M. Scarpino, *Motors for Makers: A Guide to Steppers, Servos, and other Electrical Machines*. Indianapolis, Indiana: Que, 2016.

[2] W. Tong, *Mechanical design of electric motors*. Boca Raton: CRC Press, 2014.

[3] A. Hughes and W. Drury, *Electric Motors and Drives, 4th Edition*. Newnes, 4 ed., February 2013.

[4] National Electric Manufacturers Association (NEMA), "Motors and Generators." Standards Publication MG1-2016, 2016.

[5] I. M. Gottlieb, *Practical Electric Motor Handbook*. Oxford, UK: Newnes, 1997.

[6] S. Ghosh, *Electrical Machines*. Delhi: Dorling Kindersley India, 2 ed., 2012.

[7] D. W. Spitzer, *Variable Speed Drives Principles and Applications for Energy Cost Savings*. New York, N.Y.] (222 East 46th Street, New York, NY 10017): Momentum Press, 4th ed. ed., 2012.

[8] R. Rifaat, "Industrial Motor Protection [History]," *IEEE Industry Applications Magazine*, vol. 18, pp. 10,11, November 2012.

[9] Department of Energy, "10 C.F.R. 431, Subpart B - Electric Motors." `www.ecfr.gov/cgi-bin/text-idx?SID=1d4a3e47894c42e30b45a27277fbdf5d&mc=true&node=sp10.3.431.b&rgn=div6`, 2019. Accessed: 2019-07-05.

[10] Department of Energy, "10 C.F.R. 431, Subpart X - Small Electric Motors." `www.ecfr.gov/cgi-bin/text-idx?SID=ff7f131ef3415ead7976d82050477c6f&mc=true&node=sp10.3.431.x&rgn=div6`, 2019. Accessed: 2019-07-05.

[11] M. J. Assante and R. M. Lee, "The Industrial Control System Cyber Kill Chain," Tech. Rep. 36297, SANS Institute, October 2015.

[12] O. Alexander, "ICS ATT&CK Framework: Adversary Tactics and Techniques (S4x19)." `www.brighthubengineering.com/commercial-electrical-applications/78579-determining-causes-for-electric-motor-failure/`, January 2019. Accessed: 2019-07-05.

[13] B. E. Strom, J. A. Battaglia, M. S. Kemmerer, W. Kupersanin, D. P. Miller, C. Wampler, S. M. Whitley, and R. D. Wolf, "Finding Cyber Threats with ATT&CK™-Based Analytics," Tech. Rep. MTR170202, The MITRE Corporation, Annapolis Junction, Maryland, June 2017.

[14] M. Rocchetto and N. O. Tippenhauer, "On Attacker Models and Profiles for Cyber-Physical Systems," in *Computer Security – ESORICS 2016* (I. Askoxylakis, S. Ioannidis, S. Katsikas, and C. Meadows, eds.), (Cham), pp. 427–449, Springer International Publishing, 2016.

[15] joan2937, "pigpio." `github.com/joan2937/pigpio/`, 2019. Accessed: 2019-07-15.

[16] Broadcom, "BCM2835 ARM Peripherals." `www.raspberrypi.org/wp-content/uploads/2012/02/BCM2835-ARM-Peripherals.pdf`. Accessed: 2019-07-05.

[17] Gordon, "WiringPi." `github.com/WiringPi/WiringPi`, 2018. Accessed: 2019-07-15.

[18] S. Monk, "servosix." `github.com/simonmonk/servosix`, 2019. Accessed: 2019-07-15.

[19] D. Quarta, M. Pogliani, M. Polino, F. Maggi, A. M. Zanchettin, and S. Zanero, "Breaking the Laws of Robotics - Attacking Industrial Robots," *Black Hat USA*, July 2017.

[20] R. Healey and M. Ryan, "Hacking Electric Skateboards: Vehicle Research for Mortals," *DEF CON 23*, July 2015.

[21] M. Knight and M. Newlin, "Radio Exploitation 101: Characterizing, Contextualizing, and Applying Wireless Attack Methods," *DEF CON 25*, July 2017.

[22] Z. Drias, A. Serhrouchni, and O. Vogel, "Taxonomy of Attacks on Industrial Control Protocols," in *2015 International Conference on Protocol Engineering (ICPE) and International Conference on New Technologies of Distributed Systems (NTDS)*, pp. 1–6, July 2015.

[23] S. East, J. Butts, M. Papa, and S. Shenoi, "A Taxonomy of Attacks on the DNP3 Protocol," in *Critical Infrastructure Protection III* (C. Palmer and S. Shenoi, eds.), (Berlin, Heidelberg), pp. 67–81, Springer Berlin Heidelberg, 2009.

[24] J. A. Marty, *Vulnerability Analysis of the MAVLink Protocol for Command and Control of Unmanned Aircraft*. PhD thesis, Air Force Institute of Technology, Wright-Patterson Air Force Base, Ohio, 2014.

[25] J. Valente and A. Cardenas, "Understanding Security Threats in Consumer Drones Through the Lens of the Discovery Quadcopter Family," in *Proceedings of the 2017 Workshop on internet of things security and privacy*, IoTS&P '17, pp. 31,36, ACM, November 2017.

[26] J. Daubert, D. Boopalan, M. Mühlhäuser, and E. Vasilomanolakis, "HoneyDrone: A Medium-Interaction Unmanned Aerial Vehicle Honeypot," in *NOMS 2018 - 2018 IEEE/IFIP Network Operations and Management Symposium*, pp. 1257–1262, April 2018.

[27] D. Quarta, M. Pogliani, M. Polino, F. Maggi, A. M. Zanchettin, and S. Zanero, "An Experimental Security Analysis of an Industrial Robot Controller," in *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 268–286, May 2017.

[28] B. Gorenc and F. Sands, "Hacker Machine Interface - The State of SCADA HMI Vulnerabilities," tech. rep., Trend Micro Zero Day Initiative Team, 2017.

[29] Dragos Inc., "CRASHOVERRIDE: Analysis of the Threat to Electric Grid Operations," Tech. Rep. 2.20170613, Dragos Inc., Dover, DE, June 2017.

[30] K. Fu and W. Xu, "Risks of Trusting the Physics of Sensors," *Communications of the ACM*, vol. 61, pp. 20,23, January 2018.

[31] Y. Tu, Z. Lin, I. Lee, and X. Hei, "Injected and Delivered: Fabricating Implicit Control over Actuation Systems by Spoofing Inertial Sensors," *27th USENIX Security Symposium*, August 2018.

[32] T. Trippel, O. Weisse, W. Xu, P. Honeyman, and K. Fu, "WALNUT: Waging Doubt on the Integrity of MEMS Accelerometers with Acoustic Injection Attacks," in *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*, pp. 3,18, IEEE, 2017-04.
[33] G. Zhang, C. Yan, X. Ji, T. Zhang, T. Zhang, and W. Xu, "DolphinAttack: Inaudible Voice Commands," in *Proceedings of the 2017 ACM SIGSAC Conference on computer and communications security*, CCS '17, pp. 103,117, ACM, October 2017.
[34] A. G. Finogeev and A. A. Finogeev, "Information Attacks and Security in Wireless Sensor Networks of Industrial SCADA Systems," *Journal of Industrial Information Integration*, vol. 5, pp. 6,16, March 2017.
[35] K. T. Sterne, J. M. Ernst, D. K. Kilcoyne, A. J. Michaels, and G. Moy, "Tire Pressure Monitoring System Sensor Radio Frequency Measurements for Privacy Concerns," *Transportation Research Record*, vol. 2643, no. 1, pp. 34,44, 2017.
[36] J. V. Carroll, "Vulnerability Assessment of the U.S. Transportation Infrastructure that Relies on the Global Positioning System," *Journal of Navigation*, vol. 56, pp. 185,193, May 2003.
[37] Resilient Navigation and Timing Foundation, "Prioritizing Dangers to the United States from Threats to GPS," Tech. Rep. ICSA-18-193-01, RNT Foundation, November 2016.
[38] E. Horton and P. Ranganathan, "Development of a GPS Spoofing Apparatus to Attack a DJI Matrice 100 Quadcopter," *The Journal of Global Positioning Systems*, vol. 16, pp. 1,11, December 2018.
[39] K. Zeng, Y. Shu, S. Liu, Y. Dou, and Y. Yang, "A Practical GPS Location Spoofing Attack in Road Navigation Scenario," in *Proceedings of the 18th International Workshop on mobile computing systems and applications*, HotMobile '17, pp. 85,90, ACM, February 2017.
[40] A. D. Pinto, Y. Dragoni, and A. Carcano, "TRITON: The First ICS Cyber Attack on Safety Instrument Systems," *Black Hat USA*, August 2018.
[41] H. Alemzadeh, D. Chen, X. Li, T. Kesavadas, Z. Kalbarczyk, and R. Iyer, "Targeted Attacks on Teleoperated Surgical Robots: Dynamic Model-Based Detection and Mitigation," in *Proceedings - 46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2016*, (United States), pp. 395–406, 9 2016.
[42] D. Beresford, "Exploiting Siemens Simatic S7 PLCs," *Black Hat USA*, August 2011.
[43] K. Zetter, "An Easy Way for Hackers To Remotely Burn Industrial Motors." `www.wired.com/2016/01/an-easy-way-for-hackers-to-remotely-burn-industrial-motors/`, January 2016. Accessed: 2019-07-15.
[44] T. Newman, T. Rad, and J. Strauchs, "SCADA & PLC Vulnerabilities in Correctional Facilities," tech. rep., Newman, Rad, Strauchs, July 2011.
[45] R. Langner, "To Kill a Centrifuge," tech. rep., The Langner Group, Dover, DE, November 2013.
[46] S. Kirby, "tgy – Open Source Firmware for ATmega-based Brushless ESCs." `github.com/sim-/tgy`, 2017. Accessed: 2019-07-15.
[47] Electronoobs, "My Open Source ESC." `www.electronoobs.com/eng_arduino_tut91.php`, 2019. Accessed: 2019-07-15.
[48] L. O'Donnell, "Critical Rockwell Automation Bug in Drive Component Puts IIoT Plants at Risk." `threatpost.com/critical-rockwell-automation-bug-in-drive-component-puts-iiot-plants-at-risk/143258/`, March 2019. Accessed: 2019-07-15.
[49] Cybersecurity and Infrastructure Security Agency, "ICS Advisory (ICSA-18-193-01) Eaton 9000X Drive," Tech. Rep. ICSA-18-193-01, Department of Homeland Security, July 2018.
[50] E. Kovacs, "No Patches for Critical Flaws in Fuji Electric Servo System, Drives." `www.securityweek.com/no-patches-critical-flaws-fuji-electric-servo-system-drives`, September 2018. Accessed: 2019-07-15.
[51] A. Abbasi and M. Hashemi, "Ghost in the PLC: Designing an Undetectable Programmable Logic Controller Rootkit via Pin Control Attack," *Black Hat Europe*, November 2016.
[52] C. Miller, "Battery Firmware Hacking: Inside the Innards of a Smart Battery," *Black Hat USA*, July 2011.
[53] K. Zetter, "Comedy of Errors Led To False 'Water-Pump Hack' Report." `www.wired.com/2011/11/water-pump-hack-mystery-solved/`, November 2011. Accessed: 2019-07-15.