Power-Based Intrusion Detection for Additive Manufacturing: A Deep Learning Approach

Michael Rott and Sergio A. Salinas Monroy(✉)

Wichita State University, Wichita, KS 67260, USA
MJRott@shockers.wichita.edu, Sergio.SalinasMonroy@wichita.edu

Abstract. Due to the ability of 3D-printers to build a wide range of objects at low costs, many industries are rapidly adopting additive manufacturing. However due to their sensing and communications capabilities, 3D-printers are Internet of Things (IoT) devices that are vulnerable to sophisticated cyberattacks, such as defect injection attacks. By maliciously manipulating the behavior of a 3D-printer, an attacker can compromise the integrity of a manufactured objects. To avoid detection, the adversary also compromises the sensor data reported by the 3D-printer that the operator could use to detect the attack. In this paper, we design a deep neural network that can detect such attacks by predicting the power consumption of a 3D-printer based on the object design and previous power consumption observations. By analyzing the difference between the predicted power consumption and the observed one, we can determine if the 3D-printer is under attack. By measuring the power consumption of the 3D-printer at the power line with an independent sensor, we can determine the true behavior of the 3D-printer without relying on sensor data reported by the potentially compromised 3D-printer. Compared to previous works, our proposed detection technique only requires cheap power sensors that can be easily installed. We conduct extensive experiments on a real-world additive manufacturing testbed and observe that our proposed method can detect defect injection attacks with up to 96% accuracy.

Keywords: Security · Intrusion detection · Side-channel defense · 3D-printing · Additive manufacturing

1 Introduction

Additive manufacturing (AM) is rapidly being adopted by many industries including healthcare [1,2], aerospace [3], and automotive [4]. In AM, 3D-printers manufacture objects by depositing material in a layer-by-layer fashion, which allows them to build a wide range of objects with complex geometries, and various levels of tensile strength, heat transfer, etc. Compared to traditional manufacturing where machines need tool retrofitting each time they build a new object, AM can redirect 3D-printers to produce entirely different types of objects without changing any of their parts [5]. Moreover, 3D-printers are Internet of Things (IoT) devices

L. Peñalver and L. Parra (Eds.): Industrial IoT 2020, LNICST 365, pp. 171–189, 2021.
https://doi.org/10.1007/978-3-030-71061-3_11

equipped with sensing and communications capabilities that allow factory managers to monitor and control their AM operations in real-time [6,7].

However, the use of 3D-printers to build objects for safety-critical industries such as healthcare and transportation raises major cybersecurity concerns. Sophisticated cyber attackers regularly target industrial control systems aiming to steal intellectual property or sabotage their physical processes [8,9]. Standard security controls, e.g., firewalls, intrusion detection systems, encrypted communications, etc. [10], can protect against many non-sophisticated attacks, such as an adversary stealing object designs from an unencrypted source. Unfortunately, standard security controls are inadequate to address sophisticated attacks where the adversary has enough knowledge and resources to bypass these controls. Therefore, it is crucial to design IoT-enabled AM systems that not only allow operators to enhance their systems with improved monitoring and control but also prevent malicious adversaries from compromising their operations.

A particularly serious sophisticated cyberattack against 3D-printers is the defect injection attack [11,12]. In a defect injection attacks, the adversary introduces defects into the additively manufactured objects to affect their integrity, and ultimately lead to premature (and potentially life-threatening) failure. To inject the defects, the adversary maliciously modifies the machine instructions used to build the object in such a way that it changes the manufactured object's geometric or material properties. To avoid detection, the adversary replaces the original sensor data transmitted by the 3D-printer to the operator with sensor data that appears normal. For example, the adversary can collect sensor data reported during normal operation, and then replay it during the attack while deleting the true sensor data. Such attacks can be launched by compromising the 3D-printer firmware, launching a man-in-the-middle attack, or by compromising the PC that controls the 3D-printer. Similar attacks where the adversary compromises both the physical process and the sensor data have been observed in industrial control systems, e.g., [13]. Thus, to detect defect injection attacks, we need access to information about the 3D-printer's behavior that is independent from the potentially compromised 3D-printer.

To detect defect injection attacks, researchers have proposed to analyze sidechannel signals that carry information about the behavior of the 3D-printer. By measuring the side-channel with a device that is independent of the 3D-printer, it is possible to observe the behavior of the 3D-printer and detect when it is compromised. Some works measure the acoustic emissions with microphones [14–17] to measure the movements of the 3D-printer tools. However, they need sophisticated microphones placed near the 3D-printers and can be affected by interference from other nearby machines. Researchers have also proposed to measure the power consumption of the 3D-printers to detect defect injection attacks [12,18]. Unfortunately, these works either require a complicated retrofit of the 3D-printer to measure the consumption of individual components [7], or rely on an energy consumption model [18].

In this paper, we propose to use a deep neural network to analyze the electrical power consumption of a 3D-printer using a single off-machine sensor. Specifically, our proposed deep neural network takes as input the object design and the previously observed power consumption measurements to predict the power consumption measurement. By analyzing the difference between the predicted power consumption and the observed one, we can determine if the 3D-printer is deviating from the known object design, and thus injecting defects into the object. By measuring the power consumption of the 3D-printer at the power line with an independent device, we can determine the true behavior of the 3Dprinter without relying on sensor data reported by the potentially compromised 3D-printer. Compared to previous works that employ sophisticated hardware, our proposed detection technique only requires power sensors, which can be acquired at low cost and can be easily installed on the power lines without retrofitting the 3D-printer. We conduct extensive experiments on a real-world additive manufacturing testbed. We measure the performance of our proposed detection method under different types of defect injection attacks in terms of accuracy, precision, and recall. We observe that it can successfully detect defect injection attacks with up to 96% accuracy.

The rest of this paper is organized as follows. In Sect. 2, we present the system model. Section 3 describes our use of deep learning and detection methodology. Section 4 presents the implimentation and results of our model testing. Section 5 and 6 provide a conclusion and reference to future works to be done on this subject.

2 Related Works

In this section, we provide a brief overview of works relating to additive manufacturing security.

Cybersecurity issues in additive manufacturing have received increased attention from researchers in recent years. Specifically, Yampolskiy et al. [19] describe several attacks against 3D-printers, including defect injection attacks and how they can be used to compromise the additively manufactured object's shape and material properties. Sturm et al. [20] describe void injection attacks, where the adversary creates empty spaces within the objects. This type of attack is particularly challenging to detect since it requires complex imaging such as computerized tomography (CT) scans, to observe the voids. Graves et al. [21] discuss how defect injection attacks can be used to sabotage industrial control systems and the consequences of these attacks. Do et al. [22] demonstrate how an attacker can remotely manipulate the behavior of a 3D-printer. Belikovetsky et al. [11] implement a defect injection attack that introduces voids into a 3D-printed object.

Some researchers have designed methods to detect defect injection attacks. Sturm et al. [23] propose to detect defects in a metallic additively manufactured objects by comparing its impedance to that of a control defect-free object. Unfortunately, this verification method only works with metallic objects, and must be done after building the object, which can result in expensive material waste. Wu et al. [14] propose machine learning methods that analyze acoustic and visual data from the additively manufactured objects. In [16], Belikovetsky et al. propose to compare the acoustic emissions of a 3D-printer to a recording of the same 3D-printer when it built a defect-free object. Gao et al. [17] used acceleration, visual, audio, and magnetic field measurements to create an accurate estimate of the 3D-printer's movements. The observed movements were compared to the defect-free objects to detect the attack. In [12], Belikovetsky et al. propose to monitor the power consumption of individual stepper motors inside the 3D-printer. However, these defect injection methods require the installation of multiple sophisticated sensors near, or inside, the 3D-printer, and thus are expensive, difficult to implement, and subject to interference from nearby machines. In particular, the method proposed in [12] requires opening the 3D-printer and separating the wires that feed electrical power to each stepper motor inside the 3D-printer. This poses a significant challenge to system operators since they could have multiple machines, and disassembling them could be cumbersome, complicated, and costly.

Salinas et al. [18], propose to identify defect injection attacks in a system of multiple 3D-printers by observing their aggregate power consumption with a single power consumption sensor installed on the power lines that feed the machines. However, this approach only focuses on isolating the compromised machine and assumes that an energy consumption model already exists.

In this work, we design a deep neural network mechanism that can detect defect injection attacks and only requires a single power consumption sensor that is cheap and can be easily installed on the power lines feeding the 3D-printer. These power measurements are independent from the potentially compromised measurements reported by the 3D-printer.

3 System Model

In this section, we develop a model that describes the operation of a 3D-printer, and describe our considered threat model.

3.1 3D-printer Operation Model

We consider a 3D-printer that is controlled by a PC. The 3D-printer operates an extruder head mounted on a gantry with three degrees of freedom. The planes are denoted by x, y, and z, respectively, as shown in Fig. 1. To build the objects, the extruder head deposits material on the printing bed. The 3D-printer creates objects from the bottom up in a layer by layer fashion. A set of tool-path instructions transmitted from the controller PC to the 3D-printer determines the extruder head's speed and direction of movement. Besides the tool-path instructions, the controller PC sends several build parameters to the 3D-printer, such as the temperature of the printing bed, speed of material extrusion, cooling fan speed, etc. Figure 1 shows the planes of movement of the extruder head.

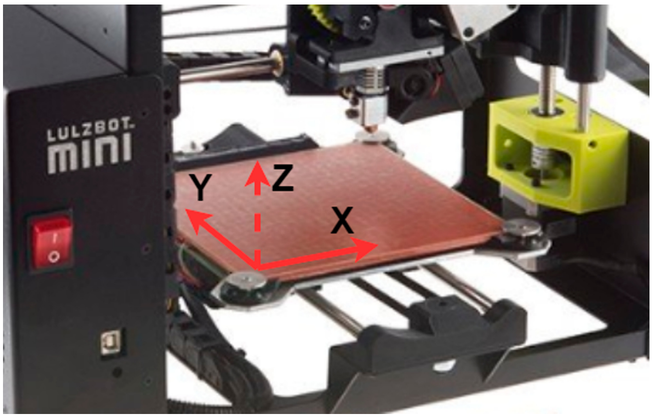


Fig. 1. 3D-printer axes.

The 3D-printer implements the tool-path instructions by actuating a set of stepper motors that, in turn control the position of the extruder head and the material deposition. Let $m_0^x, m_0^y, m_0^z, m_1^z, m_0^e$ denote the motors of the 3D-printer. Motor m_0^x and motor m_0^y move the extruder head along the x and y axes, respectively. Both motor m_0^z and m_1^z drive the motion of the extruder head along the z-axis. Motor m_0^e controls a gear that feeds material into the extruder head.

Besides the stepper motors, the 3D-printer also controls two heaters and two cooling fans. The first heater is used to maintain a constant temperature at the baseplate to help keep the print in place as well as to aid in the bonding of layers [24]. The second heater melts the material filament as it is fed through the extruder head. The cooling fans are placed near the extruder head to promote bonding between layers by reducing the temperature of the deposited material [17].

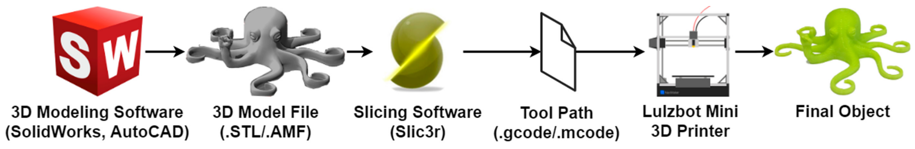


Fig. 2. 3D printing process chain.

3.2 Tool-Path Instruction Model

The 3D-printer builds objects by following tool-path instructions that specify the path its extruder head needs to follow and the amount of material it needs to deposit. To find the tool-path instructions, the controller PC first captures the surface geometry of the three-dimensional design in a stereolitography (STL) file. Next, a slicing software, e.g., Slic3r, takes the STL file as input and outputs the tool-path instructions for the 3D-printer. We show the work flow to find tool-path instructions in Fig. 2.

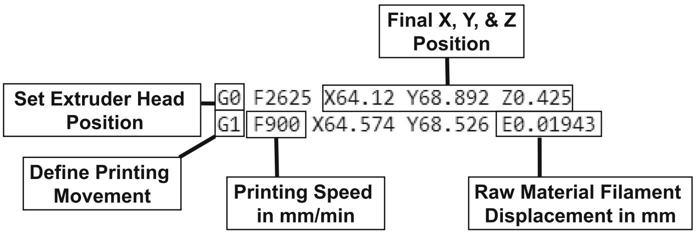

Final X, Y, & Z Position
Set Extruder Head Position
G0 F2625 X64.12 Y68.892 Z0.425
G1 F900 X64.574 Y68.526 E0.01943
Define Printing Movement
Printing Speed in mm/min
Raw Material Filament Displacement in mm

Fig. 3. Labeled G-Code example.

The tool-path instructions can be represented in several different formats. In this work, we will focus on the G-code format. Specifically, tool-path instructions coded in the G-code format specify the end position of the extruder head, the speed of movement, and the position of the extruder gear, which determines the amount of deposited material. Besides movement, G-code instructions can also be used to set the temperature of the heaters. Figure 3 shows an example of a set of G-code tool-path instructions initialize the position of the extruder head and then move the extruder head while depositing material.

3.3 Extruder Head Movement Model

We model the movement of the extruder head based on the tool-path instructions described in Sect. 3.2. The objective is to describe the expected direction and speed of movement of the extruder head based on the tool-path instructions. Since a single tool-path instruction only specifies the final position of the extruder head, we need two consecutive tool-path instructions to determine the extruder head movement. In particular, let $P_k = \{X_k, Y_k\}$ be the final position of the extruder head specified by the kth tool-path instruction, where X_k and Y_k are the X and Y coordinates, respectively. Thus, the extruder's initial position for movement k is defined as X_{k-1} and Y_{k-1}. Moreover, the change in position of the extruder head during the kth tool-path instruction along the X and Y axes is given by:

$$\delta X = X_k - X_{k-1}, \quad \delta Y = Y_k - Y_{k-1} \tag{1}$$

The movement of the raw material filament through the extrusion head can be similarly modeled. Let E_k be the final position of the extruder filament after the kth instruction. Then, the length of the raw material filament that is extruded by the 3D-printer is given by:

$$\delta E = E_k - E_{k-1} \tag{2}$$

Moreover, the speed at which the raw material filament moves during the kth tool path instruction is usually directly specified by the G-code command. We denote it by F_k. If F_k is not specified, we use the speed specified in the most recent tool-path instruction where the speed was specified.

Note that the extruder head moves along the Z-axis to raise the gantry after a layer is finished. However, the movement along the Z-axis requires the operation of two-stepper motors instead of one as it is the case with movement along the X and Y planes. Since operating two motors results in a vastly different power consumption, we leave the study of the Z-axis movements for future work.

3.4 Power Consumption Measurement Model

To build objects, the 3D-printer transforms alternating current (AC) from its power source to direct current (DC) power, which is then routed it to the step-per motors. Previous works, e.g., [12], attempt to detect defect injection attacks by measuring the power consumption of the individual stepper motors, which requires disassembly of the 3D-printer. Instead, to avoid retrofitting of the 3Dprinter, we directly measure the overall AC power consumption of the 3D-printer using an AC current sensor that is placed between the 3D-printer and the AC power source.

Since the AC current consumed by the 3D-printer follows a sinusoidal shape whose amplitude varies proportionally to the total power consumed by the 3Dprinter, we can use information about the magnitude of the peaks and valleys to determine the power consumed to execute specific tool-path commands. Specifically, let $|I_k^p| = [i_k^1, i_k^2, ..., i_k^N]$ be the vector of absolute peak AC current samples taken by the sensor during tool-path instruction k, where N is the total number of samples. Then, the average AC current peak during tool-path instruction k is given by:

$$\bar{I}_k^p = \frac{1}{N} \sum_{j=1}^{N} i_k^j \tag{3}$$

We note that although the average peak AC current $\bar{I}_k^p$ discards the non-peak current values, our experiments show that using $\bar{I}_k^p$ provides enough information for our deep neural networks to effectively detect the defect injection attacks.

3.5 Threat Model

We consider an adversary that aims to introduce a defect into the additively manufactured objects by maliciously modifying the original tool-path instructions. To avoid detection, the adversary intercepts the sensor measurements taken by the 3D-printer and replaces them with sensor measurements that match the

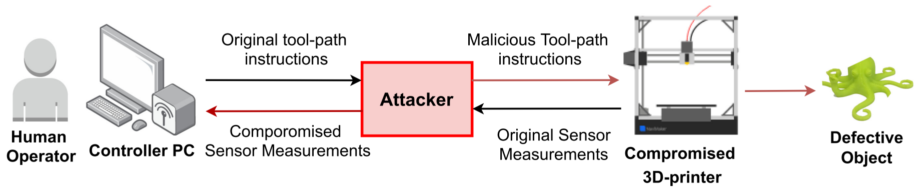

Human Operator
Controller PC
Original tool-path instructions
Comporomised Sensor Measurements
Attacker
Malicious Tool-path instructions
Original Sensor Measurements
Compromised 3D-printer
Defective Object

Fig. 4. A 3D-printer architecture under a defect injection attack.

tool-path commands of the original object design. To this end, the adversary can launch one of several cyberattacks, including compromising the controller PC or the firmware installed the 3D-printer. Moreover, the adversary seeks to avoid detection by choosing to inject defects into the manufactured object that are difficult to notice by visual inspection, e.g., a small modification to the dimensions of the object. This threat model is shown in Fig. 4

The adversary can maliciously modify the original tool-path instructions in several ways. In this work, we consider the following tool-path instruction modification attacks

1. **Insertion Attacks.** In this type of attack, the adversary inserts one or more additional tool-path instructions into the machine code that is executed by the 3D-printer. The inserted command could be solely movement, or it could also contain a command to deposit material with movement. This attack introduces defects by printing material in unintended locations or by manipulating the start position of the extruder for the next tool-path instruction.
2. **Deletion Attacks.** The adversary launches a deletion attack by removing one or more tool-path instructions from the machine code executed by the 3D-printer. This attack can introduce a defect by removing commands which print material at specific locations, or set the extruder position for the next instruction.
3. **Reordering Attacks.** In this attack, the adversary swaps the order in which two tool-path instructions are executed. The instructions need not be consecutive. The swapped instructions lead to defects during the swapped commands as well as the commands that follow, as the starting location of the extruder head for those commands will now be modified.
4. **Void Injection Attack.** In this attack, the adversary prevents the 3Dprinter from depositing material at certain positions and layers in such a way that a cavity is introduced into the printed object. Voids lead to a modified cross-section of the printed object, which will affect the stresses and strains that object undergoes. This could ultimately lead to an object with less physical integrity than the intended print [11].
5. **Printing Speed Attack.** In a printing speed attack, the adversary changes the printing speed parameter of the 3D-printer. This attack can result in a raw material filament pressure change that causes an alteration in its diameter as it is extruded. This can affect the surface morphology and integrity of the printed object [25].

We assume the adversary does not compromise any other devices such as the AC current sensor that we use in our proposed detection method. We also assume that the operator can perform destructive testing on multiple manufactured objects to collect power measurements while the printer builds objects when there is no attack. This ensures that enough power consumption measurements are available to train our proposed deep neural network.

4 Power-Based Deep Learning Attack Detection Method

In this section, we describe the proposed deep learning attack detection methodology.

To detect defect injection attacks in real-time, we propose to design a deep neural network that takes the original tool-path instructions as input to output a prediction about the power consumption of the 3D-printer. By comparing the power consumption predicted by a deep neural network to the power consumption observed by a sensor installed on the power line that feeds the 3D-printer, we can determine if the 3D-printer is deviating from the original tool-path instructions. If the predicted and observed peak current consumptions differ by more than the threshold, then we conclude that the printer is implementing a tool-path instruction different from the original tool-path instruction used as input to the deep neural network. We show our overall defect injection attack approach in Fig. 5.

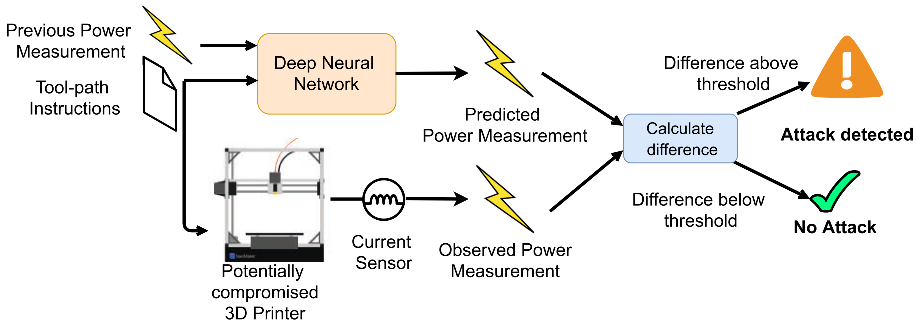

Previous Power Measurement
Tool-path Instructions
Deep Neural Network
Predicted Power Measurement
Potentially compromised 3D Printer
Current Sensor
Observed Power Measurement
Calculate difference
Difference above threshold
Attack detected
Difference below threshold
No Attack

Fig. 5. Proposed attack detection procedure.

In the rest of this section, we explain in detail each step of our proposed defect injection detection scheme.

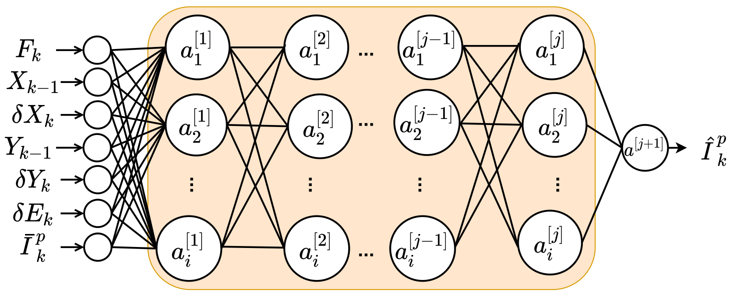


Fig. 6. Neural network architecture.

4.1 Deep Neural Network Architecture

We first present a deep neural network that predicts the power consumption of a 3D-printer while it executes a specific tool-path instruction. The deep neural network follows a multi-layer perceptron feedforward architecture as shown in Fig. 6.

As described in Sect. 3.4, we use the average peak AC current $\bar{I}^p_K$ to measure the overall power consumption of the 3D-printer. Thus, our deep neural network generates a prediction of the average peack AC current using the tool-path instruction characteristics specified in the G-code command as described in Sect. 3.3. Specifically, it uses as input the extruder head speed F_k, the previous position of the extruder head given by X_{k-1} and Y_{k-1}, the change in position δX_k and δY_k, the changes in the position of the raw material filament δE_k, and the average peak AC current from the previous tool-path instruction $\bar{I}^p_k$.

The input values are given to the input layer, which calculates the inputs to the first hidden layer, and so on until they reach the output layer. In particular, the computations of the first hidden layer are defined as follows:

$$\mathbf{a}^{[1]} = \sigma\big(\mathbf{w}^{[1]} \cdot \mathbf{x}\big) \tag{4}$$

where $\mathbf{x} \in \mathbb{R}^{7\times 1}$ is a vector that contains the input values, $\mathbf{a}^{[1]} \in \mathbb{R}^{i\times 1}$ is the output of first hidden layer, and $\mathbf{w}^{[1]} \in \mathbb{R}^{i\times 7}$ is the matrix of weights for the first layer, and σ is the sigmoid activation function.

The hidden layer is fully connected to another hidden layer of the same size i. The computations performed by the following hidden layers are defined as:

$$\mathbf{a}^{[j]} = \sigma\big(\mathbf{w}^{[j]} \cdot \mathbf{a}^{[j-1]} + \mathbf{b}^{[j]}\big), \forall j = 2 \ldots, j, \tag{5}$$

where $\mathbf{a}^{[j]} \in \mathbb{R}^{i\times 1}$ is the output of jth hidden layer, $\mathbf{w}^{[j]} \in \mathbb{R}^{i\times i}$ and $\mathbf{b}^{[j]} \in \mathbb{R}^{i\times 1}$, are the matrices of weights and biases for the jth layer respectively, and i is the size of the jth hidden layer. This hidden layer is connected to another hidden layer up to j layers.

Finally, the last hidden layer j is fully connected to a dense output layer with one node. The computations performed by the last layer are as follows:

$$\bar{I}_k^p = \sigma\Big(\mathbf{w}^{[j+1]} \cdot \mathbf{a}^{[j]} + \mathbf{b}^{[j+1]}\Big) \tag{6}$$

where $\hat{I}_k^p$ is the final scalar output of the network and represents the predicted average AC peak current for tool-path instruction k.

The ability of the deep neural network to accurately predict $\hat{I}_k^p$ depends on the parameters $\mathbf{w}^j$ and $\mathbf{b}^j$ (for all j). To find these parameters, the network is trained using average peak AC current samples taken when the printer is building an object without defect as we later describe in Sect. 5.3.

4.2 Detection of Abnormal Average Peak Current

To detect defect injection attacks against the 3D-printer, we propose to compare the predicted average peak current $\hat{I}_k^p$ to the observed one $\bar{I}_k^p$. Specifically, we calculate the square error loss function for each tool-path instruction used to build an object [26], i.e.,

$$D_k = (\hat{I}_k^p - \bar{I}_k^p)^2 \tag{7}$$

for all k. We then compare D_k to a threshold value t to determine if there is a defect injection attack. If D_k is less than the threshold value (for any k), we conclude the printer is following the original tool-path instructions used as input to the deep neural network. Otherwise, we conclude it is executing different tool-path instructions, and thus it is under attack.

5 Experimental Evaluation

In this section, we implement our proposed defect injection attack detection mechanism and evaluate its performance.

5.1 Testbed Implementation

To closely replicate a real-world 3D-printing scenario, we used a general-purpose PC connected via USB to a LulzBot Mini 3D printer, as shown in Fig. 7. To take AC load current measurements, we clamped a YHDC SCT-013 current sensor around a single wire coming from the 120V power source (i.e., the wall outlet) into the printer. This current sensor outputs voltage measurements that are linearly proportional to the magnitude of the current flowing through the wire. The output voltage resolution of the sensor is .033V/1A. We used an oscilloscope to observe the voltage output of the current sensor and collected the measurements with the general-purpose PC. In practice, the controller PC would be different from the PC collecting the measurements. However, to simplify the control of the 3D-printer and the measurement collection, we used the same PC for both functions.

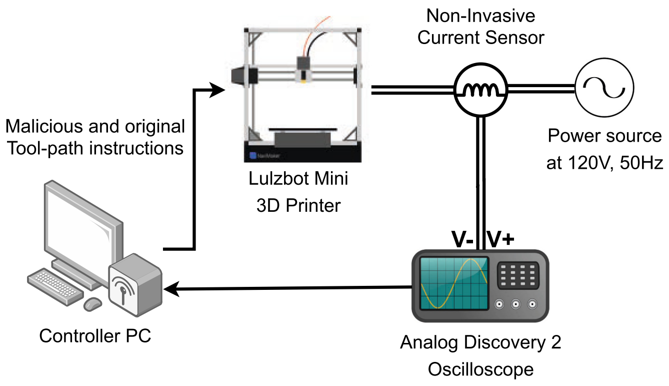


Fig. 7. Diagram of our experiment testbed for defect injection attack detection.

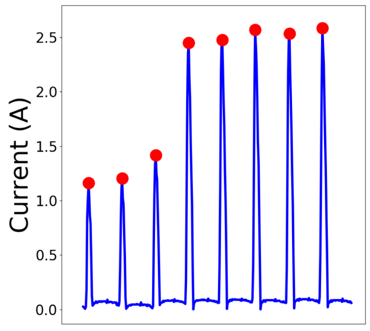
To control the 3D-printer and collect measurements, we developed a Python application that sends tool-path instructions for a specific object design to the printer, collects AC current measurements from the sensor. We also use this application to implement the defect injection attacks by changing the tool-path instructions as described in Sect. 3.5. We also developed an application that processes the AC current measurements and implements the deep neural network detection mechanism. We use PyTorch version 1.4, an open-source machine learning library, to train and implement the deep neural network. Both applications are run on a general-purpose PC with an Intel i7-7700HQ CPU, 16 GB RAM, and an NVIDIA GeForce GTX 1050 graphics card.

5.2 Power Consumption Measurement Preprocessing

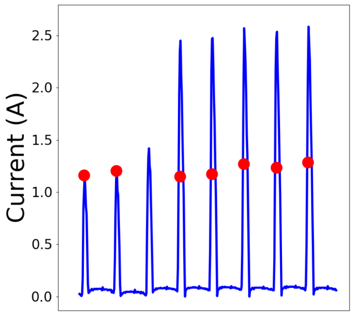
Before we can use the current consumption measurements to train the deep neural network, we need to preprocess the raw AC current measurements. Specifically, we first extract the peaks from the AC current measurements to form the vector I_p^k as described in Sect. 3.4. Since some of the peaks stored in I_p^k may correspond to the operation of the extruder heater, which does not contribute information about the movement of the extruder head, we remove peaks that are greater than a threshold. The threshold can be easily set due to the peaks corresponding to the heater being several mA higher than the peaks that correspond to stepper motors. After removing the peaks due to the heater, we can calculate the average peak AC current $\bar{I}_k^p$ as described in (3). We show an example of raw and preprocessed data in Fig. 8

5.3 Deep Neural Network Training

To train the deep neural network, we collected average peak current measurements of the 3D-printer while executing different types of tool-path instructions. Specifically, we first built the test object in Fig. 9 and collected the average peak



(a) Raw Peaks.



(b) Preprocessed Peaks.

Fig. 8. Effects of preprocessing.

current measurement for each of its 84 tool-path instructions. This process was repeated 10 times. This test object results in tool-path instructions that contain both single-plane instructions, i.e., movements parallel to one of the planes, and double-plane instructions where the extruder head moves along two planes at the same time, i.e., diagonal movements. The length of the movements required by the tool-path instructions varies between .707 and 105.8 mm. We then paired the pre-processed power consumption measurements with their corresponding tool-path instructions to create a training data set for the deep neural network described in Sect. 4.1. Table 1 summarizes the types of tool-path instructions in our data set.

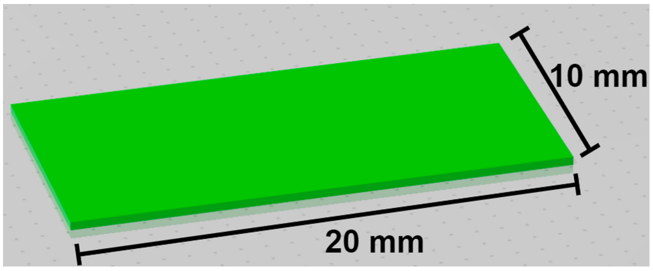


Fig. 9. 3D-model of the test object.

After building the training data set, we train the deep neural network described in Sect. 4 in Pytorch. We used the ADAM optimizer algorithm to find the parameters of the deep neural network. The loss function was set to the mean square loss, with a learning rate of .0015.

Moreover, we also build a testing data set that we later use to analyze the performance of our deep neural network. Specifically, we build the test object 50

Table 1. Types of tool-path instructions for the test object.

Instruction type	Number of appearances
No Movement	1
X-axis only	30
Y-axis only	14
X, Y-axis	39

times under each defect injection attacks described in Sect. 3.5, and measured the average peak current of the 3D-printer for each tool-path instruction.

5.4 Hyperparameter Tuning

To find the deep learning network architecture that could best predict the power consumption of the 3D-printer, we measured the performance of the neural network under several hyperparameter combinations. Specifically, for each network configuration, we re-trained the deep neural and measured the training loss, i.e., how well it predicts the average peak current, and the training time. We tested several networks with a varying number of hidden layers, and a varying number of neurons in each layer between 25 and 100.

We show the results in Fig. 10 and observe that the best performing architecture has 2 hidden layers of 100 neurons.

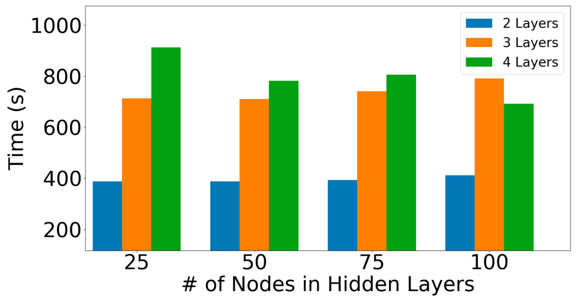


Fig. 10. Training loss under best performing hyper-parameters.

We also measured the training time for each network model tested. Figure 11 shows the training time of the proposed neural network for varying number of layers and layer sizes. We see that adding layers generally increases the training time, while the size of the layers has little effect.

Based on the training loss and training time results above, we chose a deep neural network with 2 hidden layers of 100 neurons to detect defect injection attacks in our experiments.

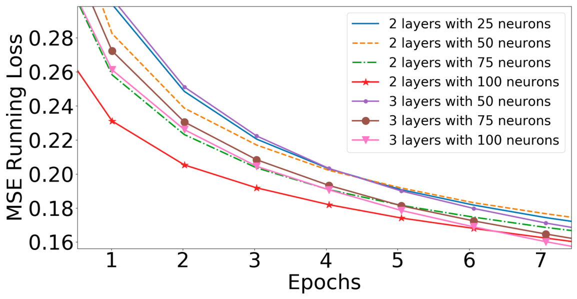


Fig. 11. Training time under varying parameters.

5.5 Results

We first evaluate the performance of the proposed deep neural network in detecting the attacks described in Sect. 3.5 under varying values for threshold t. As described in Sect. 4.2, after calculating the the square error value in (7), we need to compare it to a threshold t. Figure 12 shows the accuracy of the deep neural network to identify attacks for varying values of the threshold t. We see that choosing a threshold of .73 gives a high accuracy for all attack types.

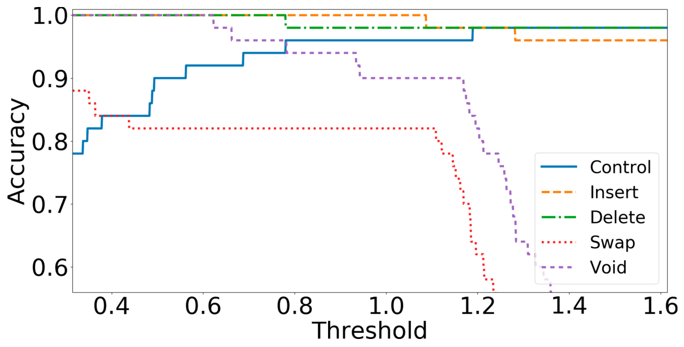


Fig. 12. Attack detection accuracy.

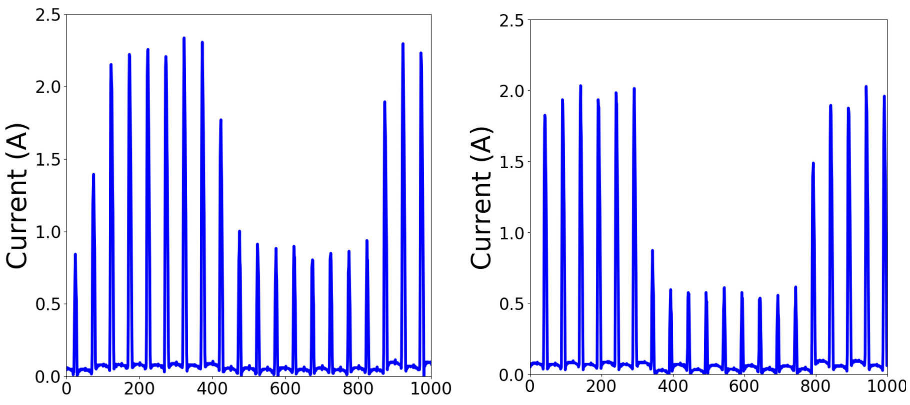
Table 2 shows the performance of the proposed detection scheme when the threshold $t = 0.7$ for each of the attacks. The first column shows the type of attack launched against the test object or if it was a defect-free print. The second column indicates the number of times our detection scheme correctly classified the test object print as either defect-free or compromised. The third column shows the number of types it incorrectly classified the object print. Columns four, five, and six show the accuracy, precision, and recall of the proposed detection scheme under each type of attack. We observe that the deep neural network is able to correctly detect the insertion and deletion attacks in all 50 trials of the test object printing.

We also observe that the reorder attacks can be correctly detected with an accuracy of 82%, precision of 95%, and recall of 82%. The detection scheme shows even better performance for void attacks with an accuracy, precision, and recall of 98%, 96%, and 98%, respectively. The last row shows that the proposed method only had two false positives out of 50 defect-free prints.

Table 2. Detection accuracy. Threshold = .73.

Attack	Correct	Misclassified	Accuracy	Precision	Recall
Insertion	50	0	1.00	0.96	1.00
Deletion	50	0	1.00	0.96	1.00
Reorder	41	9	0.82	.95	.82
Void	49	1	0.98	.96	.98
Control Print	48	2	0.96	NA	NA

The reason why our deep neural network has a high-accuracy in detecting the insertion and deletion attacks is that they result in high differences between the predicted and observed average peak current not only during the affected tool-path instruction but through the entire object build. This allows the neural network to have more opportunities to observe an abnormal average peak current. Under a reorder attack, our deep neural network also achieves a high accuracy but has fewer opportunities to detect the abnormal average peak current since only the two swapped instructions will result in abnormal readings. In void attacks, the extruder motor stops running, which results in differences between the predicted and observed values. We show differences caused by void attacks in Fig. 13.



(a) Movement with Extrusion. (b) Movement without Extrusion.

Fig. 13. Effects of void attack

6 Conclusions and Future Work

In this paper, we have investigated the problem of detecting defect injection attacks against 3D-printers. Since the adversary can compromise the sensor measurements reported by the potentially compromised 3D-printer, we use an external sensor that measures the power consumption of the 3D-printer. We then use a deep neural network that takes as input the object design and the previously observed power consumption measurements and predicts the power consumption measurement for the current tool-path instruction. If the difference between the predicted power consumption and the observed one is large, we can determine that the 3D-printer is deviating from the known object design, and injecting defects into the object. Previous works use sophisticated hardware that is difficult to install and often requires the operator to disassemble the 3D-printer. In contrast, our proposed detection technique only requires a low-cost power sensor that can be easily installed without retrofitting the 3D-printer. Our extensive experimental evaluations show that the proposed method can detect several defect injection attacks with up to 96% accuracy.

In the future, we plan to investigate the application of our proposed attack detection method in substractive manufacturing. Similarly to AM, substractive manufacturing processes design files to produce G-Code or M-Code commands that define the movement of the machine's tools. However, instead of depositing material in specific locations as in AM, substractive manufacturing uses CNC machines to shape blocks of raw material into the manufactured parts using a number of tools (drills, saws, etc.). Our proposed method could be used to monitor CNC machine power consumption, and detect malicious variations in tool-path commands. Additionally, we plan to imporove the accuracy of the system to more closely monitor indivudal stepper motors.

References

1. Dodziuk, H.: Applications of 3d printing in healthcare. Kardiochirurgia i torakochirurgia polska = Polish J. Cardio Thoracic Surg. **13**(3), 283–293 (2016)
2. Sandström, C.: Adopting 3D printing for manufacturing - the case of the hearing aid industry. In: Ratio Working Papers 262, The Ratio Institute, December 2015. https://ideas.repec.org/p/hhs/ratioi/0262.html
3. Wood, J.: 3d printing's soaring impact on aviation and aerospace. Forbes, September 2019. http://www.forbes.com/sites/mitsubishiheavyindustries/2019/09/24/3d-printings-soaring-impact-on-aviation-and-aerospace/#1b3d57a7269c
4. Sculpteo: automotive and 3d printing: the complete guide to the 3d printed car (2020). https://www.sculpteo.com/en/3d-learning-hub/applications-of-3d-printing/3d-printed-car/
5. Shahrubudin, N., Lee, T., Ramlan, R.: An overview on 3d printing technology: technological, materials, and applications. Procedia Manufact. **35**, 1286– 1296 (2019). http://www.sciencedirect.com/science/article/pii/S2351978919308169, the2nd International Conference on Sustainable Materials Processing and Manufacturing, SMPM 2019, 8-10 March 2019, Sun City, South Africa

6. Tao, F., Cheng, Y., Xu, L.D., Zhang, L., Li, B.H.: CCIoT-CMfg: cloud computing and internet of things-based cloud manufacturing service system. IEEE Trans. Ind. Inf. **10**(2), 1435–1442 (2014)
7. Buckholtz, B., Ragai, I., Wang, L.: Cloud manufacturing: current trends and future implementations. J. Manufact. Sci. Eng. **137**(4), 040902 (2015)
8. Verizon: Data breach digest (2019). http://www.verizonenterprise.com/resources/breports/rp_data-breach-digest-2017-perspective-is-reality_xg_en.pdf
9. U.S. Department of Homeland Security: ICS-CERT monitor-incidence response activity (2015)
10. Stouffer, K.: Guide to industrial control systems (ICS) security. NIST Spec. Publ. **800**(82), 16 (2011)
11. Belikovetsky, S., Yampolskiy, M., Toh, J., Gatlin, J., Elovici, Y.: dr0wned – cyber-physical attack with additive manufacturing. USENIX Association, Vancouver, BC, August 2017. https://www.usenix.org/conference/woot17/workshop-program/presentation/belikovetsky
12. Gatlin, J., Belikovetsky, S., Moore, S.B., Solewicz, Y., Elovici, Y., Yampolskiy, M.: Detecting sabotage attacks in additive manufacturing using actuator power signatures. IEEE Access **7**, 133421–133432 (2019)
13. Langner, R.: Stuxnet: dissecting a cyberwarfare weapon. IEEE Secur. Priv. **9**(3), 49–51 (2011)
14. Wu, M., Song, Z., Moon, Y.B.: Detecting cyber-physical attacks in cybermanufacturing systems with machine learning methods. J. Intell. Manuf. **30**(3), 1111–1123 (2019). https://doi.org/10.1007/s10845-017-1315-5
15. Song, C., Lin, F., Ba, Z., Ren, K., Zhou, C., Xu, W.: My smartphone knows what you print: exploring smartphone-based side-channel attacks against 3d printers. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, CCS 2016, pp. 895–907. Association for Computing Machinery, New York (2016). https://doi.org/10.1145/2976749.2978300
16. Belikovetsky, S., Solewicz, Y.A., Yampolskiy, M., Toh, J., Elovici, Y.: Digital audio signature for 3d printing integrity. IEEE Trans. Inf. Forensics Secur. **14**(5), 1127–1141 (2019)
17. Gao, Y., et al.: Watching and safeguarding your 3d printer: online process monitoring against cyber-physical attacks. Proc. ACM Interact. Mob. Wearable Ubiquit. Technol. **2**(3) (2018). https://doi.org/10.1145/3264918
18. Monroy, S.A.S., Li, M., Li, P.: Energy-based detection of defect injection attacks in IoT-enabled manufacturing. In: 2018 IEEE Global Communications Conference (GLOBECOM), pp. 1–6 (2018)
19. Yampolskiy, M., Schutzle, L., Vaidya, U., Yasinsac, A.: Security challenges of additive manufacturing with metals and alloys. In: Rice, M., Shenoi, S. (eds.) ICCIP 2015. IAICT, vol. 466, pp. 169–183. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-26567-4_11
20. Sturm, L.D., Williams, C.B., Camelio, J.A., White, J., Parker, R.: Cyber-physical vulnerabilities in additive manufacturing systems: a case study attack on the.stl file with human subjects. J. Manufact. Syst. **44**, 154–164 (2017). http://www.sciencedirect.com/science/article/pii/S0278612517300961
21. Graves, L.M.G., Lubell, J., King, W., Yampolskiy, M.: Characteristic aspects of additive manufacturing security from security awareness perspectives. IEEE Access **7**, 103833–103853 (2019)
22. Do, Q., Martini, B., Choo, K.R.: A data exfiltration and remote exploitation attack on consumer 3d printers. IEEE Trans. Inf. Forensics Secur. **11**(10), 2174–2186 (2016)

23. Sturm, L., Albakri, M., Williams, D.C.B., Tarazaga, D.P.: In-situ detection of build defects in additive manufacturing via impedance-based monitoring, pp. 1458–1478 (2016)
24. Spoerk, M., Gonzalez-Gutierrez, J., Sapkota, J., Schuschnigg, S., Holzer, C.: Effect of the printing bed temperature on the adhesion of parts produced by fused filament fabrication. Plast. Rubber Compos. **47**(1), 17–24 (2018). https://doi.org/10.1080/14658011.2017.1399531
25. Geng, P., et al.: Effects of extrusion speed and printing speed on the 3d printing stability of extruded peek filament. J. Manufact. Process. **37**, 266–273 (2019)
26. Kravchik, M., Shabtai, A.: Detecting cyber attacks in industrial control systems using convolutional neural networks. In: Proceedings of the 2018 Workshop on Cyber-Physical Systems Security and PrivaCy, pp. 72–83 (2018)