Security Implications of Malicious G-Codes in 3D Printing

Jost Rossel[*], Vladislav Mladenov[†], Nico Wördenweber[*], Juraj Somorovsky[*]

[*]*Paderborn University. {jost.rossel, juraj.somorovsky}@upb.de, n.woerdenweber@paderborn.com*
[†]*Ruhr University Bochum. vladislav.mladenov@rub.de*

Abstract

The rapid growth of 3D printing technology has transformed a wide range of industries, enabling the on-demand production of complex objects, from aerospace components to medical devices. However, this technology also introduces significant security challenges. Previous research highlighted the security implications of G-Codes—commands used to control the printing process. These studies assumed powerful attackers and focused on manipulations of the printed models, leaving gaps in understanding the full attack potential.

In this study, we systematically analyze security threats associated with 3D printing, focusing specifically on vulnerabilities caused by G-Code commands. We introduce attacks and attacker models that assume a less powerful adversary than traditionally considered, broadening the scope of potential security threats. Our findings show that even minimal access to the 3D printer can result in significant security breaches, such as unauthorized access to subsequent print jobs or persistent misconfiguration of the printer. We identify 278 potentially malicious G-Codes across the attack categories Information Disclosure, Denial of Service, and Model Manipulation. Our evaluation demonstrates the applicability of these attacks across various 3D printers and their firmware. Our findings underscore the need for a better standardization process of G-Codes and corresponding security best practices.

1 Introduction

The rapid advancement and widespread adoption of 3D printing technology have significantly transformed manufacturing, prototyping, and even personal fabrication. From aerospace components [5] to medical devices [37, 42] and consumer goods [1], 3D printing enables precise and on-demand creation of complex objects directly from digital models. Recent reports value the market at around 27.52 billion US dollars in 2024, reaching a value of 150 billion by 2032 [19, 44]. However, using 3D printing introduces new security challenges, as any manipulation of the printing instructions can have serious implications for the integrity and safety of the final products.

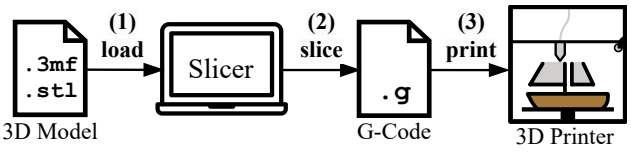


Figure 1: General workflow when using a 3D printer, showing the process from digital model to physical object.

Known Threats in 3D Printing. A typical 3D printing process includes the 3D model (e.g., STL [7] or 3MF [2]), the slicer application, which translates the model into printing instructions, and the 3D printer itself (see Figure 1). Each of these components can be a target for attacks.

Researchers have demonstrated that abusing these components makes it possible to introduce flaws into 3D-printed objects. For example, altering the digital model or the G-Codes can lead to structural failures that are difficult to detect but are potentially dangerous [10, 20, 38, 51, 62, 67]. Additionally, attacks have been shown to manipulate the G-Code files before being transmitted to 3D printers, resulting in compromised final products that do not meet their intended specifications [38]. These attacks highlight the potential risks associated with 3D printing and the need for robust security mechanisms to protect against such threats.

Gaps in Prior Work (§3). Despite the growing body of research on the security of 3D printing, gaps remain. Notably, prior work has not provided a systematic analysis of malicious G-Codes, which are the fundamental commands used to control the 3D printing process. Instead, most studies have employed G-Codes as a means without investigating the full range of threats they pose. Furthermore, the attacker models commonly used in previous research are often limited to specific cases, for example, assuming a person-in-the-middle attacker [38], a virus on a victim's machine, or malicious firmware [39, 43]. While these attacker models are important, they do not encompass the entire spectrum of potential attacks. In particular, less powerful adversaries were not investigated.

Novel Attacker Models (§4). We introduce three novel attacker capabilities that assume a less powerful adversary, challenging the existing focus on highly capable attackers. For instance, we show that an attacker could steal the G-Codes of the subsequent print jobs by only using the 3D printer once. Moreover, the attacker could either damage or lock the 3D printer by using a simple one-line command.

Systematic Threat Analysis of G-Codes (§5). In contrast to previous works, which utilized specific G-Codes for selected attacks, we take a systematic approach by analyzing all possible G-Codes and assessing their potential for malicious usage. To the best of our knowledge, this is the first comprehensive approach of its kind. We identified 278 of 593 potentially malicious G-Codes, which we organized into three distinct categories: Information Disclosure, Denial of Service (DoS), and Model Manipulation.

New Attacks (§6). Based on the systematic evaluation, we created novel attacks using broadly supported G-Codes. Such attacks include spying on arbitrary print jobs, locking the 3D printer, or misconfiguring it. Each attack is sorted into the novel attacker capabilities.

Evaluation (§7). We show the applicability of the attacker capabilities through practical evaluations, uncovering security vulnerabilities across eight 3D printing devices. The tested devices cover a broad range of different firmware projects and use cases, and confirm the generality of our attacks. Based on our findings, we initiated a responsible disclosure process. We have submitted our report to the respective firmware and hardware vendors and the national CERT team.

Artifacts. We provide all artifacts of our security analysis.[1] This includes all analyzed G-Codes, the source code that generates applicable test cases, and the testing protocols, including information about the printer, firmware, notes, and photos of the printed objects. Additionally, we provide a tool for users to assess how vulnerable their printer is.

Contributions. Our main contributions are

- a systematization of related work regarding G-Codes which highlights gaps in existing research (Section 3),
- three novel attacker models that operate under less powerful conditions, expanding the scope of potential security threats in 3D printing (Section 4),
- a systematic discovery of G-Codes, their categorization based on their potential for malicious use, and analysis of their usage patterns in real-world 3D models (Section 5),
- a detailed description of proof-of-concept attacks showing the applicability of the new attacker model (Section 6), and
- an evaluation across eight 3D printing devices, determining their real-world applicability (Section 7).

2 Foundations of 3D Printing

3D printing, or Additive Manufacturing, involves fabricating a three-dimensional object by incrementally adding material layer by layer. In contrast, traditional Subtractive Manufacturing methods, like a CNC mill, carve out the desired object by cutting away material from a solid block, such as aluminum. Various types of 3D printers use different materials and techniques for fabrication. Among the existing technologies, Fused Filament Fabrication (FFF) is the most prevalent [40].[2] Depending on the technologies and the manufacturer of the 3D printer, different instructions are needed to control the printer. Often, these instructions are G-Codes [20].

The general workflow of creating a single object using a 3D printer that operates on G-Codes is shown in Figure 1. First, the user creates or obtains a digital 3D model file (1), for example, by using a CAD program and exporting it to an appropriate format. Typical file types are STL [7], OBJ [60], AMF [27], and 3MF [2], which are specially designed to be used with 3D printing [50]. The model (1) is then turned into printing instructions (2) that the printer understands. This process is called *slicing* as the 3D model is divided into horizontal slices and is typically done through dedicated software called a *slicer*. The slicer is often also used to add supporting structures to the 3D model so it can be printed, and to arrange multiple objects for the same print. The slicer's printing instructions depend on both the software itself and what the printer understands and requires. Depending on the setup and hardware of the printer, the printing instructions can be transferred manually via a USB stick or SD card, or via a serial or network connection. In the latter case, the printer can often be controlled by the slicer, and the instructions are never persistently stored on the printer itself (e.g., on an SD card). No higher-level protocols to secure the communication with the 3D printer (e.g., TLS) are widely adopted.

2.1 Fused Filament Fabrication (FFF)

As G-Codes are typically used in conjunction with FFF printers [49] and FFF printers are the most commonly used type of printer [40, 52], we limit our scope to this type of printer. In FFF, a string of plastic called *filament* is fed through a hot nozzle on a print surface to form a cross-section of a 3D model. After the first layer is done, the 3D printer moves the nozzle up and continues to print the next cross-section of the model, hence fabricating the object by fusing the layers of the filament. The stepper motor that presses the filament through the nozzle is called the *extruder*. The *hot end* is the commonly used term to describe the combined parts of the print head

1. see `https://doi.org/10.5281/zenodo.14719309`

2. The market forecast lists "FDM" (Fused Deposition Modeling), which is a trademark of Stratasys, Inc. [59], we will be using the term FFF coined by the RepRap project [48] as a non-trademarked alternative for the same manufacturing process.

that get hot when the respective heater is turned on. These parts are the heater cartridge, the heat block, the nozzle, and the lower part of the heat break. The term *hot end* is also often used synonymously to nozzle. The *print bed* is the surface on which the nozzle puts down the first layer of the molten filament. Many 3D printers have a heated print bed to help with the adhesion of the print.

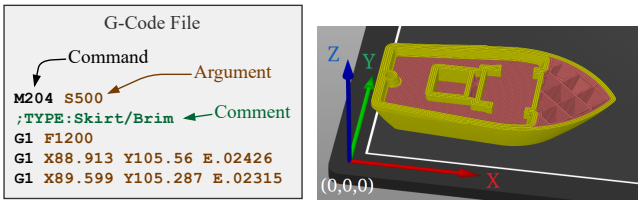
A printer needs a stepper motor for each of the three movement axes `X`, `Y`, and `Z` (see Figure 2b). The exact way these axes are moved depends on the printer. Additionally, the extruder is considered a fourth axis `E`. The *feedrate* is the speed at which any of these axes are moved.

The printer's movable parts are controlled by the *board*, which is typically a printed circuit board (PCB) with a microcontroller, stepper motor controllers, and any other required connectivity to the printer and interfaces to the user. Often the boards include serial ports to interact directly with the firmware. The *firmware* interprets incoming instructions and acts according to them. In the realm of FFF printers, Marlin and RepRap are popular open-source firmware projects. Many consumer FFF printers use a version of Marlin that is adapted by the manufacturer. An example of this are the Creality printers we tested (cf. Table 2).

2.2 G-Code

In general, a G-Code is a single instruction for a computer-controlled machining tool. The smallest unit of a G-Code is a *word*; a word is a letter followed by a number, for example, `X1`. The letter is called *address character* [61]. The first word in a G-Code is the command; all the remaining are its arguments (see Figure 2a). For example, `G1` in line 4 of Figure 2a tells a 3D printer to move the print head in a straight line to the given coordinates and to extrude the given amount of material. G-Codes can reuse their argument values for the next commands [31]. For example, `G1` in line 3 of Figure 2a defines how fast the movement should be done (i.e., a feedrate of 1200 mm/min). This value is reused for all upcoming `G1` commands until it is changed again. The combination of multiple G-Codes forms a program that produces an object; we call this a *G-Code file*. While various address characters can be used,[3] the most common ones are `G` and `M`.[4] Besides the different possible letters, the language family is generally known as "G-Code" [61].

G-Codes are the dominant control language for CNC machines in subtractive manufacturing [61]. They were first standardized in 1963 as EIA RS-274 [29] and have since been extended in multiple other standards [6, 15, 26, 31]. Outside the standardized G-Codes, there are also "vendor-specific" G-Codes and extensions allowing programming constructs like loops, which—while not standardized—are used by multiple manufacturers. These were added to allow the format to control more complex CNC machines that the original standard was not designed for. While the G-Codes used in 3D



(a) G-Code Example (b) G-Code Rendering and Axis

Figure 2: G-Code example showing the nomenclature (a) and a rendering showing a partial print (of [13]) with visible printing lines and typical axis labels and origin (b).

printing generally follow the idea of the early CNC G-Codes, they have been extended and modified to fit the needs of 3D printing. For example, the `E` axis to control the extruder is not part of the G-Code standards. Some projects are compliant to some standard [49], and others are only partially or not at all compliant [32]. Many of these 3D printing specific G-Codes are documented in the Wiki of the RepRap project [49]. But even this list is not exhaustive, as some manufacturers use their own G-Codes [24]. The G-Codes supported by a printer, thus, depend on the printer's firmware—also called a G-Code *flavor*—and the hardware of the printer, as typically only the G-Codes are available that the hardware supports. Overall, while G-Codes are standardized for basic usage in subtractive CNC machines, these standardized versions are rarely used, leading to a patchwork of different variants.

3 Related Work

Attacks with Malicious G-Codes. A lot of security research in 3D printing has used G-Codes either directly or as a part of larger attacks [10,20,38,62,67]. One often seen method of using G-Codes for malicious purposes is to change them to introduce defects in the printed object [8, 10, 57, 62, 67]. To do so effectively, the attacker needs full exposure to the G-Code file. The means to achieve this attack vector include (1) changing the printer's firmware [39, 43], (2) installing malware on the victim's computer [10], or (3) intercepting the G-Codes while they are transferred to the printer [38].

Rais et al. [46], Pearce et al. [43], and Moore et al. [39] all attacked the firmware running on the 3D printer. Moore et al. [39] created a malicious Marlin firmware that could be triggered by specific commands. One attack is a "[subverted] program control" [39] where the printer ignores the incoming G-Codes and instead prints a hard-coded model; the other attack increases the extruded material. Rais et al. [46] not only presented nine attacks using malicious firmware, but they also classified attacks on 3D printing firmware into attacker goals.

3. ISO 6983 [26], for example, defines 23 different letters.

4. It is not clear what the "G" stands for, some [55,65] state that it stands for "geometry", others [31] for "general". "M" is the "miscellaneous" category of commands [61].

Their attacks included classic vectors on the model itself, one affecting the air quality of the room the printer is in, and an attack in which the printer creates an object that is later used to physically destroy parts of the printer itself. Unlike the other two papers, Pearce et al. [43] did not target the firmware. The authors created a malicious bootloader that accesses the RAM location where the Marlin firmware is storing the G-Codes that are to be printed and edits them to introduce defects. Due to the limited storage space available, attacks must be simple; two attacks (at roughly 1600 and 1200 bytes large) are presented. *Material Reduction* changes the G-Codes so that commands that normally extrude material while moving instead move without extruding, and *Material Relocation* changes at which point the print material is extruded. Both attacks reduce the strength of the material.

Belikovetsky et al. [10] presented a method to sabotage 3D prints by modifying the G-Codes through malware. This worm searches for G-Code files and adds gaps in the print by changing the G-Codes. Similarly, Turner et al. [57] used a virus that altered the G-Codes to introduce defects in the printed object when writing the G-Code file to a USB stick.

Moore et al. [38] analyzed desktop applications for 3D printing and found that the communication between the application and the printer is not secure. This allows an attacker to intercept and modify the G-Codes.

Other Attacks and Attack Channels. Yampolskiy et al. [63] provided an overview of security research regarding 3D printing. They highlighted several attacks from the collected studies, such as model manipulation attacks (e.g., with the goal of breaking a printed model during its usage), data exfiltration attacks (e.g., with the goal of performing (industrial) espionage), and various availability attacks. The works targeting 3D printing systems presented in recent years achieved these goals using different techniques. For example, using side-channel attacks to reconstruct 3D models [12] from acoustic [4], power [21], or mobile phone [22,53] sensor data, or attacking the network capabilities that some printers have [16,35]. Researchers also considered direct attacks using the 3D model files. In 2017, Sturm et al. analyzed the impact of maliciously modified STL files and potential means for detecting such STL modifications [54]. In 2023, Rossel et al. showed that it is possible to perform Data Exfiltration, Model Manipulation, and Denial of Service attacks by exploiting the structure of 3MF files [51].

Countermeasures. The importance of the attacks modifying printed models, which a human operator might not notice [57], fostered research concentrating on the detection of malicious changes [3, 8, 11, 20, 64, 66, 67]. There are variants of these countermeasures using acoustic signals, electric current, thermal cameras and sensors, cameras, CT scans [66], or a combination of these [3].

Beckwith et al. used a red-team blue-team approach to detect malicious changes to G-Codes [8]. The red team manipulates the print by changing the G-Codes, and the blue team tries to detect these changes. Belikovetsky et al. [9] used audio signals from a 3D printer to create a digital signature, which detects modifications to the G-Code by identifying changes in the signature. Gao et al. [20] proposed a monitoring approach to detect cyber-physical attacks on 3D models. Blocklove [11] used an FPGA-based intermediary to detect and simulate attacks on 3D printers, while Ahsan et al. [3] applied security and Quality Assurance (QA) techniques to identify anomalies in 3D models through measurements.

Gaps in Prior Work. Despite the interest in the security of 3D printing, several gaps remain in the existing literature. Firstly, prior work does not provide a comprehensive and systematic evaluation of malicious G-Codes; G-Codes are solely used to facilitate the attacks. Secondly, the attacker models considered in previous studies are constrained by assumptions, such as the presence of a network attacker, installed malware, or malicious firmware. They mostly assume an attacker that has full control over the G-Code file. This leaves a gap in understanding the full range of potential threats posed by G-Code manipulation or injection. Finally, much of the research has concentrated on attacks involving Model Manipulation, such as altering the 3D model to introduce defects while leaving other potential attack classes underexplored. These gaps underscore the need for a systematic security evaluation of 3D printing, considering a broader range of attack vectors and threats posed by malicious G-Codes.

4 Attacker Model

Our attacker model makes assumptions about how the victim acts, what the attacker wants to achieve, and what capabilities the attacker has.

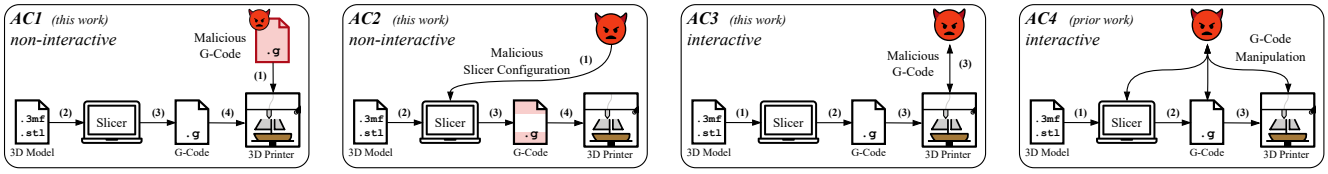
Victim. The victim is the user of a 3D printer. The printer might be company or personal property or publicly accessible, for example, in a Makerspace or through an online printing service.

Attacker Goals. Based on the recent works (e.g., [51, 63]), we consider the attacker successful if at least one of the following goals is met:

Model Manipulation: Apply physical changes to the printed model. This tends to weaken the structural integrity of the model by changing parameters for the targeted print.

Information Disclosure: Extracting sensitive data to the attacker. This includes data about prints and the printer, but also data about or from the user.[5]

5. This is also called "Intellectual Property Theft", "Data Exfiltration", or "Surveillance" in the related work.



(a) The attacker uses the 3D printer and injects malicious G-Codes. The next printing jobs are affected.

(b) The attackers manipulate the slicer on the victim's machine to affect the next printing jobs using this slicer.

(c) The attacker injects malicious G-Codes during the printing of the target model.

(d) The attacker modifies G-Codes before they are being processed by the 3D printer.

Figure 3: An overview depicting the attackers' capabilities.

Denial of Service: Affecting the availability of the printer. This includes physical harm to the victim, the 3D printer, or its environment.

We assume that both Model Manipulation and Information Disclosure target a specific print; we call this the *target object*.

Attacker Capabilities (AC). The capabilities change depending on the timeframe and the scope at which an attacker can access the G-Code. For example, they might only be able to access the printer for a short time and can only execute some G-Codes before the next print. We thus define the Attacker Capabilities (AC) on what the attacker can change in the executed G-Codes on an abstract level. The definitions are illustrated and motivated through real-life applicability scenarios. We order the resulting ACs from the weakest (AC1) to the strongest attacker (AC4). The ACs are depicted in Figure 3, and their abilities regarding a target print's G-Code file are shown in Figure 4.

AC1 assumes the weakest attacker, who only requires short access to the printer. This can be achieved if they print before the victim on a public printer and add the malicious codes at the end of their own command sequence, or have temporary access to the printer and can execute some G-Codes by inserting an SD card and executing a file.

This AC is motivated by G-Codes that can cause misconfigurations, impacting future prints. While many configurations set via G-Codes do not persist through a shutdown, an attacker can work around this limitation by using `M500` to permanently store a configuration in EEPROM (Electrically Erasable Programmable Read-Only Memory). Such configuration persists when the printer's power is turned off and is re-loaded after a reboot, affecting all prints until the configuration is changed again. Moreover, an attacker can achieve persistence through a firmware update. An update from an SD card can be triggered by `M997`. This can allow an attacker to flash a custom firmware image and effectively gain arbitrary code execution.

> ***AC1:*** *The attacker can inject G-Codes before the G-Codes of the target object are executed. They cannot change the G-Codes used for printing the target object itself.*

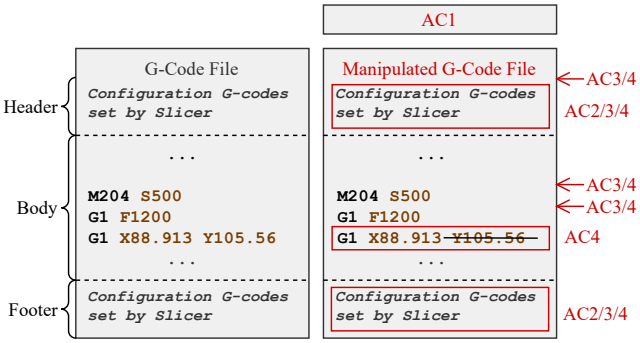


Figure 4: Depending on the AC, the attacker can manipulate the entire G-Code file (of a target print) or only parts of it. The file does not have to exist as such on the printer; it could be transferred interactively. AC2 can influence G-Codes in the header and footer (defined by the slicer). AC3 can inject G-Codes while the file is printing. AC4, the strongest attacker, can modify any content.

The injection at fixed points in the G-Code file (i.e., AC2) can mainly happen if the attacker gets access to the slicer's configurations. These configurations typically include custom G-Code instructions placed between the slicer-generated header (for printer setup) and the main body of the G-Code file (where the majority of the movement commands are). These custom instructions are typically not restricted by the slicer. If the attacker can access these configurations directly, the resulting manipulations are included in every future print until the attack gets noticed. Alternatively, the attacker can provide custom configuration instructions directly in their project files; reading project files with configuration instructions is supported, for example, by Prusa Slicer. If these files get shared on a marketplace like Thingiverse, the printed model would include the attacker's manipulations and influence configurations of 3D printers processing them.

> ***AC2:*** *The attacker can inject G-Codes at fixed points within the G-Codes of the target object by affecting the configuration used during G-Code generation. The fixed points are at the beginning and end of the G-Codes sent to the printer.*

An AC3 attacker can inject G-Codes while the target object is being printed and might observe the responses to the injected G-Codes. This is, for example, possible if the attacker can inject the sent G-Codes "on-the-fly", for example, through a breached network protocol or serial connection. For a remote attacker, this could be possible through a connected web interface that can send G-Codes, like Octoprint [23].

> ***AC3:*** *The attacker can inject G-Codes while the target object is printed. They might be able to observe the responses of the printer to their malicious G-Codes.*

AC4 models the strongest attacker, which has been considered in prior works. This attacker has full control over all commands of the targeted print. This is the only capability that allows the attacker to read, remove, add, and modify *any* (existing) G-Codes. Full access to all G-Codes can be possible if the attacker has malware running on the system where the model is sliced [10], uses a virus to change the G-Code file that is transferred via USB [57], or controls the bootloader [43] or firmware [39] of the 3D printer. In general, any technique that allows the attacker to rewrite G-Codes would fall under AC4.

> ***AC4:*** *The attacker can read, remove, add, or modify any G-Codes of the target print.*

Both AC1 and AC2 are non-interactive, meaning the attacker cannot read the G-Codes of the target object and does not get a direct reply from the printer. An attacker in AC3 or AC4, on the other hand, could be able to read the reply from the printer or the G-Codes of the target print.

5 Understanding the Impact of G-Codes

In this section, we discuss our approach towards systematizing G-Codes and analyzing them regarding their security impact. From an attacker's perspective, it is necessary to know (1) which G-Codes exist and which are supported by prominent firmware, (2) which malicious G-Codes exist, and (3) how to increase the probability of successful attacks. We answer these questions in the following sections.

5.1 Systematization of G-Codes

G-Code is the essential language driving 3D printers, but its interpretation and usage vary depending on the firmware. Understanding these variations is crucial for ensuring consistent print quality and enhancing cross-compatibility. For instance, in RepRap, the `G29` command performs basic bed leveling, whereas, in Marlin, it initiates a more complex, multipoint leveling sequence requiring multiple parameters.

Our analysis involves two key approaches: documentation review and real-world G-Code analysis. For both collections, artifacts are available.[6]

Documentation Review. Collecting the G-Codes documented by various projects and manufacturers [14, 18, 24, 30, 33, 34, 36, 45, 47, 49], we found a total of 593 unique codes, some of which might have different meanings when used on printers using different firmware. We combined them into one document, including all the descriptions from the sources, where they came from, and which firmware supports them according to the documentation.

Real-world G-Codes. To understand the real-world usage of different G-Code commands, we gathered G-Code files generated using the default configurations from various slicers and 10 000 G-Code files directly from Thingiverse, an online marketplace for 3D models.[7]

To obtain G-Code files as they are produced by slicers, we installed eight freely available slicers (as seen in Table A.4) and sliced the 3DBenchy [13] model for each pre-configured printer model included in the slicers. This is achieved through GUI scripting, as most slicers do not have a command-line interface. Overall, this yields 464 G-Code files.

Thingiverse is an online marketplace where people upload their creations, called *Things*, for others to use. For each Thing, the creator can upload different files. Usually, these are files to store 3D models, like STL [7], or project files from the 3D modeling or CAD programs, but in some circumstances, the creators also upload the already sliced G-Code file. As the Thingiverse API is rate-limited to 300 calls in 5 minutes, there is no feasible way to get all G-Code files attached to the more than 6.6 million Things. We, thus, relied on an older available data set of the metadata [50] and random sampling from all Things. Overall, we obtained 10 000 G-Code files from Thingiverse.

Documentation Coverage. In combination, we obtained 10 464 files that represent a real-world usage of how G-Codes are used. Within these files, we found 344 unique G-Codes ("used" G-Codes). Of the 344 codes, only 226 are defined by documentation (see Figure 5). As we cannot determine the behavior of all G-Codes that are missing documentation, our further analysis focuses solely on the documented codes.

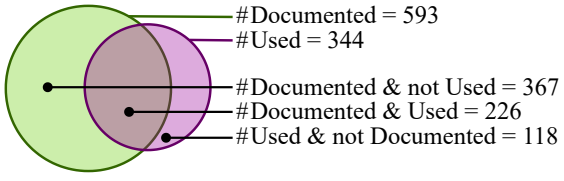


Figure 5: A discrepancy between documented and used G-Codes shows that 226 out of 711 G-Codes are both documented properly and used in the wild.

6. see `https://doi.org/10.5281/zenodo.14719309`
7. see `https://www.thingiverse.com`

5.2 Dangerous G-Codes

We analyzed all 593 G-Codes that we have information on, meticulously extracting G-Codes that are potentially dangerous. Through this extensive review, we identified and systematized the various threats associated with these G-Codes, categorizing them into the three attacker goals. To further deepen our understanding of these threats, we conducted a manual analysis on a theoretical level, examining the implications of each category in detail and assessing their potential impact on 3D printing security.

Figure 6 includes the amount of G-Codes per each category, where each G-Code can be part of more than one. The full list can be seen in Table A.5. Overall, 278 of the 593 codes can be considered potentially dangerous.

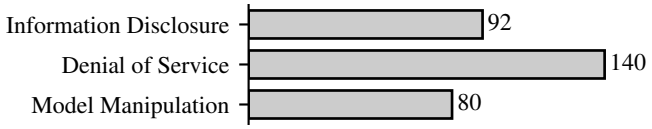


Figure 6: Number of G-Codes in each attack category.

5.3 G-Codes Usage Patterns

Each G-Code is used differently in the context of a G-Code file or print job. For instance, some G-Codes related to configurations are executed only at the beginning or the end of a print job. Conversely, some G-Codes occur multiple times throughout a file. The position of a G-Code influences the AC this particular G-Code can be used in. For example, if an attack uses a G-Code, which typically appears *only* at the beginning of a file, an attacker with AC2 can overwrite it. An attacker with AC1 cannot exploit this G-Code since they can only inject G-Codes in front of the G-Code file. If a G-Code is used throughout a file, the attacker needs to continually (re-)set the value and, thus, requires at least AC3.

To determine the usage patterns of G-Codes, we analyzed the collected data and extracted the relative positions of each G-Code in all files in our dataset. This can be visualized through histograms that show the usage of various G-Codes within all 10464 analyzed files. For each file, the lines where the command occurs are added. The position in the file of the line is normalized, where 0 is the start of the file and 1 is the end. The histograms use 100 bins, so each line is 1 % of a file. Using these histograms, we can determine which capabilities the attacker requires for an attack based on that G-Code.

For example, Figure 7a shows that the extruder temperature is typically only changed at the very beginning and end of a G-Code file. This means the attacker in AC2 can set a malicious extruder temperature, which stays valid for the entire printing process. On the other hand, the fan speed is regularly changed throughout a file (see Figure 7b). Thus, injecting a malicious `M106` demands permanent re-setting of the value, requiring an attacker with at least AC3.

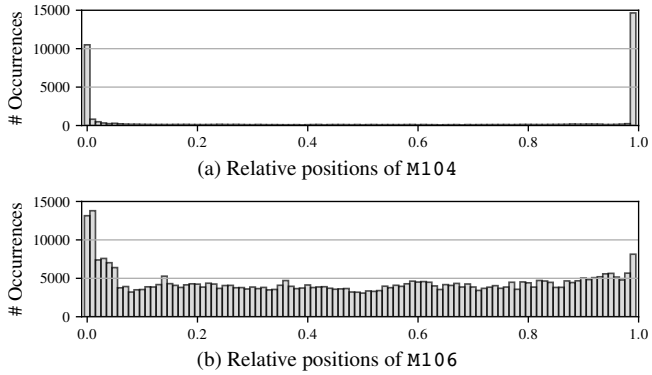


Figure 7: Histograms of the positions where `M104` and `M106` are used over all files.

6 Attacks

Due to the high amount of possible attacks identified in Section 5.2, it is infeasible to evaluate all of them on the myriad of possible 3D printers, all of which support a different subset of G-Codes. The unavailability of specific G-Codes can influence the attacker in which a specific attack is executed, or even prevent attack executions completely. To reduce the data to a set of attacks that we can feasibly evaluate, we (1) collected *specific* attacks from the related work and manually evaluated available G-Codes, (2) created *proof-of-concept* attacks using broadly supported G-Codes, and (3) matched them into their ACs by applying the usage data (Section 5.3).

The attacks presented should be regarded as *proof-of-concept*, as their effectiveness depends on the specific printer and its setup. In particular, AC1 and AC2 rely heavily on the slicer configuration. If the slicer generates G-Codes that override values set by the attack, the attack may fail. Consequently, the G-Codes provided here for a specific attack may not function with every slicer and 3D printer combination.

The results can be seen in Table 1. Please note that the example G-Codes are symbolic representations of the attacks in that category, indicating what the attack is based on. For example, "`G1`; ~~`G1`~~; `G1`" indicates that a movement command has been deleted by an attacker. Especially, the amounts the `X/Y/Z/E` axes would move are simplified. Only related work regarding G-Codes is listed. The related work might use the same type of attack for a vulnerability, describe it for potential detection, or similarly mention or explain the concept.

6.1 Information Disclosure

Information Disclosure via G-Codes is a novel attack class. It describes attacks where the attacker learns information about the print, the printer, or the user that they should not know. We divided this goal into *Intellectual Property Theft*, where an attacker obtains the G-Codes of a previous or upcoming print, and *Metadata Leakage*, where the attacker obtains metadata about the current print or the printer.

Table 1: G-Code attacks on 3D printers, showing what attacks are possible assuming weaker attacker capabilities.

	Novel Contributions					Related Work (AC4)	
Attack	**# Malicious G-Codes**	**AC1**	**AC2**	**AC3**	**Example**	**Source**	**Example**
Information Disclosure 6.1	**92**						
├ Intellectual Property Theft	3	●	●	●	`M928 log.g`	—	—
└ Metadata Leakage	90	●	●	●	`M115`	—	—
Denial of Service 6.2	**140**						
├ Interrupt Printing 6.2	123						
│ ├ Infinite Loop	14	○	●	●	`M808`	—	—
│ ├ Delay Commands	56	○	○	●	`G4`	—	—
│ ├ Ignore Commands / Stop Print	27	○	●	●	`M28`	—	—
│ └ Destroy Model / Make Unusable	40	○	●	●	`G20`	—	—
└ Disable Access / Bricking 6.2	30						
├ Software	18	●	●	●	`M512`	—	—
└ Hardware	12	●	●	●	`M104 S500`	[46]	`G1`
Model Manipulation 6.3	**80**						
├ Toolpath Manipulation 6.3.1	17						
│ ├ Voids	8	○	○	●	`M28, M29`	[3, 8, 10, 20, 67]	`G1; ~~G1;~~ G1`
│ ├ Infill Anomaly	7	○	○	○	—	[3]	`G1`
│ ├ Surface Anomaly (X/Y Shift)	13	○	○	●	`G1 X/Y±1`	[11]	`G1 ~~X/Y1~~ X/Y2`
│ ├ Layer Height Anomaly (Z Shift)	11	○	○	●	`G1 Z±1`	[3, 11]	`G1 ~~Z1~~ Z2`
│ ├ Layer Height	6	○	○	○	—	[20]	`G1 ~~Z1 E1~~ Z2 E2`
│ └ Print Angle	6	○	○	○	—	[3, 67]	`G1`
├ Faulty Extrusion 6.3.2	44						
│ ├ Under Extrusion	21	●	●	●	`M200 D1.5`	[3, 8, 11]	`G1 ~~E2~~ E1`
│ ├ Over Extrusion	19	●	●	●	`M200 D1`	[3, 11]	`G1 ~~E1~~ E2`
│ ├ Material Relocation	6	○	○	●	`G1 E±1`	[8, 43]	`G1 ~~E1~~; G1 ~~E2~~ E3`
│ ├ Filament Retraction	19	○	○	●	`G1 E±1`	[11]	`G1`
│ └ Printing Speed	18	○	○	●	`M220 S50`	[3, 20]	`G1 ~~F2000~~ F1000`
└ Temperature Changes 6.3.3	30						
├ Bed Temperature	15	●	●	●	`M140 S30`	[3]	`M140 S30`
├ Nozzle Temperature	19	●	●	●	`M143 S30`	[3, 11]	`M104 S30`
└ Fan Speed	7	○	○	●	`M106 S0`	[3, 11, 20]	`M106 S0`

● indicates an attack exists in the category under the specific AC, ○ indicates it does not. No related work or attack is marked by —.

Intellectual Property Theft can be achieved through the `M928` command, which opens the file given as an argument on the internal storage (typically an SD card) and writes all processed commands into that file. This allows the attacker to store files that are otherwise not stored on the device but printed remotely and can be achieved with every AC. To get access to the exfiltrated data, the attacker either needs brief physical access to the printer or the firmware to support remote file downloads (e.g., RepRap [17]). If that does not work, the attacker can get the current position of the printing head using the `M114` command. As this command needs to be inserted at least between every G-Code that moves the print head to get a detailed replica of the object, this is not possible in AC1 or AC2. The `M154 Sx` command enables an automatic report of the current print head position every `x` seconds. This is less precise but allows an attacker only to send it once and not overload the printer with `M114` commands. This is possible in AC2 and higher.

For *Metadata Leakage*, various commands can provide information about the printer's state. Among those, `M115` provides information about the firmware used and `M503` about basic configurations.

6.2 Denial of Service

DoS attacks on 3D printers can be split into two different goals; in one, the attacker interrupts a single print, in the other, they make the whole printer unusable through either software or hardware bricking.

One way to disrupt printing is by causing the printer to be stuck in an *Infinite Loop* while executing G-Codes. This can be achieved through the `M808` command. This allows to set a "Goto" marker that can be jumped to. The `L` argument defines how many iterations are to be done and can be set to `0` to run infinitely. An example can be seen in Listing 1. Alternatively, an attacker can *Delay Commands* by using `G4 P1000`, which pauses the execution of commands for `P` milliseconds, or by using `M0`, which lets the printer wait until the command `M108` is sent or the user presses a button on the printer. `M220` can be used to set the printing speed to 1 % of the normal speed.

```
1 M808 L0 ; set marker to repeat forever
2 ; execute arbitrary commands (or do nothing)
3 M808 ; jump to last marker
```

Listing 1: G-Codes to create an infinite loop.

Sending `M28 log.g` causes the printer to write the G-Codes that are processed into a file instead of executing them. This can trick the printer to *Ignore Commands* sent to it.

Another approach for interrupting the print is to destroy the model itself. A simple way to achieve this is to inject a move command that drives the nozzle into the print, for example, `G1 Z-1`, but many of the codes used for Model Manipulation can be used to achieve this if their parameters are chosen correctly. Other options make the whole print unusable, for example, changing the size by switching to inches instead of millimeters for all axes (i.e., `G20`).

Via `M512`, a user can set a password for the printer, which locks all functions. No prints or configuration can be done without providing the correct password. This allows an attacker to lock out the user from their printer; the user's only option is to re-flash the printer's firmware.[8]

Using `M907` the electric current used for driving specific motors can be set. Setting this to an incorrect value can damage the motor. Possible attacks against the hardware, which many printers are protected against, are to overheat certain elements (e.g., via `M104`) or to move beyond the physical boundaries that the printer is designed for. For example, the print bed could be damaged by driving the print head into it by applying negative absolute values `G1 Z-2`. Another example is provided by Rais et al. [46] where in their "Print Your Own Grave" attack, the printer first prints an object that later can be moved around to reach beyond the physical limitations of the print head. Rais et al. use this to break the print bed made from glass by throwing it off the printer. These types of attacks can be achieved in AC1. More subtly, an attacker can increase the printing speed to increase the wear on the printer's hardware.

6.3 Model Manipulation

Model Manipulation attacks can have different goals. An often-used one is for the object to be structurally weaker, meaning it breaks more easily. This can be achieved through attacks like injecting voids into the structure, causing the layers of the print not to stick together properly, or even printing at a different angle [67]. Temperature changes can also lead to the object being weaker due to specific material properties and requirements. Changes to the surface can lead to the object not fitting in with others (e.g., due to too large dimensions) or causing problems due to too high friction (e.g., through a rougher than expected surface).

Especially for Model Manipulation, there exist various attacks from related work that can be achieved through G-Code. While some related works describe the *result* of the modification (i.e., "layer delamination"), others describe the *source* (i.e., "Z-layer shift"). In this section, we combine the attacks found in different related work [3, 8, 11, 20] according to the source of the manipulation, as these are closer correlated to attacks achievable through G-Code modifications. We divided Model Manipulation again into different types of manipulations, those targeting the toolpath (the actual movement of the print head), the extrusion, or any temperature.

6.3.1 Toolpath Manipulation

Generally, the related work has covered these attacks mainly through rewriting the existing G-Codes (AC4). Hence, Table 1 lists them using `G1` (the main movement command).

The *Voids* manipulation introduces cavities into the model's shell or infill. This can be achieved by removing commands or changing the path the print head takes. In AC3 we can create voids by using `M28` and `M29`. As discussed with the *Ignoring Commands* attack, `M28` writes commands to a log file instead of executing them. `M29` re-enables the execution of the G-Codes. By injecting these codes before and after other G-Codes they are effectively disabled. An example can be seen in Listing 2.

```
1 G1 X1 Y2 E1
2
3 G1 X2 Y3 E1
4 G1 X3 Y4 E1
5
```

(a) Original G-Codes

```
G1 X1 Y2 E1
M28 file.g
G1 X2 Y3 E1
G1 X3 Y4 E1
M29
```

(b) Attack in AC3

```
G1 X1 Y2 E1

~~G1 X2 Y3 E1~~
~~G1 X3 Y4 E1~~
```

(c) Prior Work (AC4)

Listing 2: *Voids Attack*. In (b), we show how voids are executed in AC3 via `M28` and `M29`. (c) shows the same effect by deleting G-Codes done by a stronger attacker.

For *Print Angle* and *Infill Anomaly*, the manipulation changes the angle at which the model is printed and the pattern, density, or other aspects of the model's infill, respectively. Both require changes to the movements of the printer and can only be achieved in AC4. Both can cause the model to be weaker in certain scenarios [67].

The *Surface Anomaly* and *Layer Height Anomaly* manipulations introduce `X/Y` axis and `Z` axis shifts into the normal movement of the print. The former mainly causes uneven surfaces, which leads to higher friction between parts. The latter can cause the layers to delaminate (i.e., separate). As printers usually use absolute positioning,[9] we can achieve this effect by switching to relative mode, moving the print head in the desired direction, and returning to absolute mode. An example can be seen in Listing 3. By doing this in the `X/Y` plane, we can move single points in the path of the printer by an offset and create uneven paths. A similar anomaly can be achieved for the layer height (`Z` axis). To change the "resolution" of the layers (i.e., changing the layer height and extruded material), the existing G-Codes need to be changed (*Layer Height*, AC4).

8. This assumes that no default password is set in the compiled firmware, which is the default behavior when enabling the password feature in Marlin.

9. Relative mode can have precision problems due to floating-point addition inaccuracies.

```
G91 ; switch to relative positioning
G1 X.1 Y.1 ; shift X/Y layers
G90 ; switch back to absolute positioning
```

Listing 3: G-Codes to shift layers when the printer is using absolute positions. This allows the Surface and Layer Height Anomaly attacks in AC3.

(a) Original	(b) Attack in AC3	(c) Prior Work (AC4)
`G1 X1 Y2 E1`	`G1 X1 Y2 E1`	`G1 X1 Y2 E1`
	`G1 E-1`	
`G1 X2 Y3 E1`	`G1 X2 Y3 E1`	`G1 X2 Y3` ~~`E1`~~
	`G1 E1`	
`G1 X3 Y4 E1`	`G1 X3 Y4 E1`	`G1 X3 Y4` ~~`E1`~~ `E2`

Listing 4: *Material Relocation*. This assumes the `X/Y/Z` axes to be in absolute and the `E` axis to be in relative movement mode. (c) shows the modification introduced by Pearce et al. [43] (adapted for relative extrusion). (b) shows how a similar effect can be introduced only adding G-Codes, not modifying them.

6.3.2 Faulty Extrusion

These attacks change how the material is extruded. Generally, too much material can cause parts to not fit and too little can cause the parts to be weaker (similar to *Voids*).

Over and under extrusion can be achieved by injection of commands (AC3). Assuming the extruder is in relative mode, one can add commands that extrude more material (meaning `G1` with a positive `E` value) or retract the material (negative `E` value). When retracting, the next command that wants to extrude material will not eject material for the retracted amount. This is sometimes deliberately done by the slicer when the printer intends to move without extruding material to "keep the material in", for example, before starting the next layer to print. We can change the *Filament Retraction* deliberately to affect the connection between the layers by injecting over or under extrusion commands right after the intended retraction.

The *Material Relocation* attack [43] moves material in one part of the print to another (i.e., a mixture of over *and* under extrusion). This way, checking the model by weight does not reveal the modification, which it would for *Voids*, for example. The parts where the material is removed still weaken the object. Listing 4a shows an example of a G-Code sequence and the respective modifications in AC4, as introduced by Pearce et al. [43] and our adaptation, which works in AC3. As the extruder, like the other axes, can be in relative or absolute movement mode, we might have to do the same switch to the relative mode and back as shown in Listing 3.

Over and under extrusion can also be achieved through changes to configurations before the print (AC2). The command `M221` overrides the normally extruded amount. Its argument `S` is a percentage that is multiplied onto *every* `E` axis movement. `M221 S50`, for example, sets the extruded material to 50 %.[10] Another command that can be used to cause faulty extrusion is `M200`. This enables the "Volumetric Extrusion" mode of operation. Normally, the value for the `E` axis defines the linear movement of the filament in mm, in the volumetric mode the value specifies mm^3. For this, the printer needs to know the diameter of the filament to be able to compute the volume; `M200 D1.75` would enable volumetric extrusion and set the diameter of the filament to 1.75 mm. This allows the G-Code file to be reusable when changing the filament to one with a different diameter, as the values for `E` do not need to change. This mechanic can be abused to cause under or over extrusion. Provided the slicer produces G-Codes that assume linear extrusion, changing to volumetric would cause the printer to extrude the wrong amount of material. By increasing the assumed diameter, less material is extruded, as the printer assumes more material per linear movement.[11]

Changing the *Printing Speed* at which the printer is operated can impact the bonding of the material between layers (due to improper cooling of the previous layer). To achieve this with AC4, one can change the feedrate argument of the `G1` commands. This allows speed-ups and downs at parts of the print where it is impactful. A similar effect can be achieved with a more permanent change of the printing speed using `M220`, which sets the overall movement speed of all axis motors of the printer to a percentage given by the `S` argument, or with the `M221` command for only the extruder.[12]

6.3.3 Temperature Changes

This category includes everything that changes the temperatures of the various heated elements of a 3D printer. The correct temperatures largely ensure that the material is properly extruded and bonded.

Changing the *Bed Temperature* might cause the print to fail in the early stages due to bad print bed adhesion. This could cause the print to slide around, which leads to imperfections or prints that fail to finish. Using `M140` an attacker can set the targeted bed temperature. As this code is used only in the beginning and the end of files (see Figure A.11), attacks with both AC2 and AC3 can utilize it. The attacker with AC1 can utilize G-Codes that allow setting a maximum temperature, like `M143 H0 S30`, which sets the maximum temperature of the print bed to 30 °C on printers with the RepRap firmware. An equivalent attack can be done on the *Nozzle Temperature*, which affects the printed material. The code for setting the temperature directly is `M104`, which can be utilized by an

10. We sorted this attack under AC2 as the G-Code `M221` is only used at the very beginning and end of the files (see Figure A.9).

11. `M200` is only used 15 times in the dataset, we thus sort it under AC1.

12. The `M220` code is used 1183 times in the dataset, but as it is actively used throughout a number of these files (see Figure A.10) we do not categorize it under AC2 though it is theoretically possible.

attacker with AC2, as it is mainly used at the beginning and end of files (see Figure 7a).

Changing the *Fan Speed* can have similar effects to changing the nozzle temperature, as this fan stabilizes the heating and cooling of the material. The fan's speed is typically done through `M106` a frequently used G-Code (see Figure 7b). Thus, AC1 and AC2 cannot effectively change this value.

7 Evaluation

In this section, we evaluate the proof-of-concept attacks from Section 6 against four 3D printers and four printer control boards. The exact printers and boards are shown in Table 2. The results can be seen in Table 3.

Table 2: Evaluated Devices

	Device	Firmware
Printers	Creality **Ender** 3	Marlin 2.0.8.2
	Prusa i3 MK3S	Prusa 3.13.3
	Ultimaker S3	Ultimaker 8.3.1
	innovatiQ TiQ 5	n/a
Boards	Duet3D **Duet** 2 Ethernet	RepRap 3.5.2
	Artillery Sidewinder X2	Marlin 2.0.9.1
	Creality **Ender** 3 **Pro**	Marlin Creality
	Creality CR-30 3D**PrintMill**	Marlin 2.0.9.8

The firmware information is reported according to the result of `M115`, if applicable. Highlighted parts of names are used to refer to the device. All devices are evaluated in their default configuration.

7.1 Testing Methodology

We are not testing attacks from related work that have already been shown to work with full G-Code control (i.e., AC4) as, by definition, these will work on every printer if adapted to the available G-Codes. All attacks depend on the printer supporting the used G-Codes, which depends on the firmware and its configuration. For each device, we tested the default firmware and configuration.

Depending on the firmware, which can be open source or proprietary, different G-Codes for different tasks are supported. It is impossible to test all firmware variants, we, thus, selected popular and diverse devices, covering all major open-source firmware projects.

For the test of many of the Model Manipulation attacks, we have implemented a framework that creates a single print testing multiple attacks (or variants of attacks) in one run. This framework takes a normally sliced file of small cubes and modifies each cube with a different attack. A few base layers are always printed normally before the attack is applied. This allows us to see the impact on the model. We do not test the actual impact these attacks have on the model, as various previous works have already shown how the strength of the model differs (e.g., [10, 43, 46, 54, 56, 67]). For this evaluation, we deem a Model Manipulation attack successful if we see a *visual* impact or if the behavior needed for the attack can be seen. For the visual changes to the model, we often parameterize the attacks with very high or low values that a stealthy attacker would not choose.

Testing Limitations. While the attacks in the Model Manipulation and Information Disclosure categories can be typically tested without danger to the device, the attacks targeting DoS might not. In these cases, we test so that no damage is done to the printer or avoid testing an attack in full. The control boards cannot be tested on attacks that require the mechanical elements of the printer, as most Information Disclosure and DoS attacks do not require these parts, they can be tested.

For the tests of the Model Manipulation attacks on the Ultimaker and innovatiQ printers, we could not print models as we have done with the other printers, as the actual print of the models takes hours, and the printers are in frequent use at the local Makerspace that allowed us to test them. We, thus, call a printer vulnerable to an attack if it supports the G-Codes needed for it and behaves as expected for these G-Codes. Additionally, testing is limited with both printers as a direct communication channel (i.e., network or serial connection that supports G-Codes) is either disabled (innovatiQ) or not supported (Ultimaker). Attacks that rely on a back-channel are not possible with these devices. Tests were executed through loading them on an SD card, exactly like these devices are used at the local Makerspace.

7.2 Information Disclosure

An attacker can successfully extract information about the printed object and the device in six of eight cases. The exceptions are the Ultimaker and the innovatiQ printers. On all other devices, `M114` ("report position") is supported and works, while `M154` ("auto-report position") is not supported on any of the tested devices.

`M928` ("log G-Codes to file") works as expected on the Prusa printer and the Ender Pro and PrintMill boards. In the case of the Ender, although `M928` is not reported as "unsupported" by the firmware, it does not behave as expected. The printer's screen reports the filename on the display, but all following codes are neither executed nor logged. We assume this to be a bug in the firmware, as other users also came across this issue.[13] Due to the limitations mentioned in Section 7.1, attacks that require a back-channel (e.g., `M114`) are not possible on the Ultimaker and the innovatiQ printers.

7.3 Denial of Service

The infinite loop attack using the jump label code `M808` works as expected on the PrintMill board. No other tested device

13. see, for example, `https://www.reddit.com/r/3Dprinting/comments/177pe96/m28_m928_m29/`

Table 3: Evaluation of attacks on 3D printers, showing the applicability of malicious G-Codes. Ultimaker is the only 3D printer providing higher security against attacks.

Attack	Lowest AC[a]	Printers				Boards			
		Ender	Prusa	Ultimaker[b]	innovatiQ	Duet	Artillery	Ender Pro	PrintMill
Information Disclosure									
├ Intellectual Property Theft	AC1	◐ AC3	●	○	○	◐ AC3	◐ AC3	●	●
└ Metadata Leakage	AC1	●	●	○	○[c]	●	●	●	●
Denial of Service									
├ Interrupt Printing									
│ ├ Infinite Loop	AC2	○	○	○	○	●	○	○	●
│ ├ Delay Commands	AC3	●	●	◐ AC4	●	●	●	●	●
│ ├ Ignore Commands / Stop Print	AC2	●	●	○	○	●	○	●	●
│ └ Destroy Model / Make Unusable	AC2	◐ AC3	◐ AC3	◐ AC4	●	–	–	–	–
└ Disable Access / Bricking									
├ Software	AC1	○	○	○	○[c]	○	○	○	○
└ Hardware	AC1	○	◐ AC3	×	×	–	–	–	–
Model Manipulation									
├ Toolpath Manipulation									
│ ├ Voids	AC3	●	●	○	○	–	–	–	–
│ ├ Surface Anomaly (X/Y Shift)	AC3	●	●	◐ AC4	●	–	–	–	–
│ └ Layer Height Anomaly (Z Shift)	AC3	●	●	◐ AC4	●	–	–	–	–
├ Faulty Extrusion									
│ ├ Under Extrusion	AC1	●	●	◐ AC2	●	–	–	–	–
│ ├ Over Extrusion	AC1	●	●	◐ AC2	●	–	–	–	–
│ ├ Material Relocation	AC3	●	●	◐ AC4	●	–	–	–	–
│ ├ Filament Retraction	AC3	●	●	◐ AC4	●	–	–	–	–
│ └ Printing Speed	AC3	●	●	◐ AC4	○	–	–	–	–
└ Temperature Changes									
├ Bed Temperature	AC1	◐ AC2	◐ AC2	◐ AC2	◐ AC2	–	–	–	–
├ Nozzle Temperature	AC1	◐ AC2	◐ AC2	◐ AC2	◐ AC2	–	–	–	–
└ Fan Speed	AC3	●	●	◐ AC4	●	–	–	–	–

● Successful attack in lowest AC. ○ No successful attack. × Not tested to avoid harm. Attacks that are only possible in AC4 are left out of the table as they were not evaluated.
◐ Successful attack in higher (listed) AC. – Cannot be tested.

[a] The lowest possible AC of any of the contained attacks.
[b] All AC3 attacks on the Ultimaker printer are categorized as AC4, as the device does not allow direct communication via G-Codes by default.
[c] Cannot be tested due to the setup of the printer, but could be possible based on the advertised capabilities.

```
M28 loop.g; start writing to file 'loop.g'
while true
  echo "Hello World!"
M29; stop writing to file
M32 loop.g; execute file
```

Listing 5: An infinite loop using RepRap's meta commands.

supports the instruction. We found an alternate way of creating infinite loops for the Duet board using the RepRap firmware's meta commands.[14] These commands allow simple programming structures like while-loops. Using the command sequence shown in Listing 5, the printer prints "Hello World" to the serial connection indefinitely.

Making the model unusable by changing the scale only works on the innovatiQ printer, as the command is not supported on all other printers. Driving the print head into the already printed material, on the other hand, works on all tested printers. Delaying commands works on all tested devices while using `M28` to ignore incoming commands works on five out of eight. The Ender printer caps the nozzle's temperature at 260 °C; thus, no dangerous overheating is possible. The Prusa printer allows setting temperature above the hardware's limits, but the heating is aborted if these temperatures are actually reached. We did not test higher-than-recommended temperatures on the printers from the Makerspace (i.e., Ultimaker and innovatiQ). Regarding the "Print Your Own Grave" attack style of hardware DoS, no immediate fragile pieces—like a glass print bed—are in range of a printed "extension" on either of the printers. The `M907` to set motor voltages is supported by the Prusa printer. We did not test this to avoid damaging the printer, but the attack works in principle.

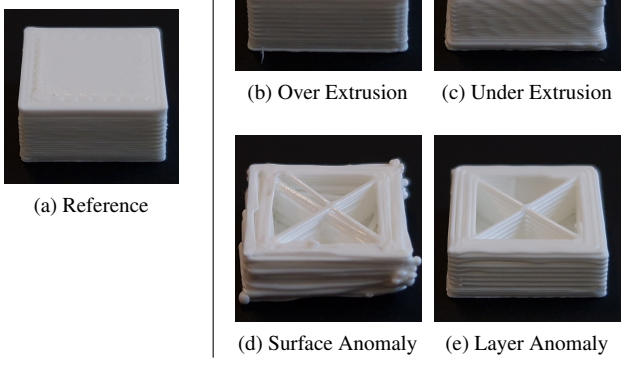
None of the devices support setting passwords by default. To test the functionality of the attack, we compiled a version of the PrintMill's firmware with the password feature enabled. This reflects how a user who wants to secure their printer would enable it. The attack worked as expected.

14. see `https://docs.duet3d.com/User_manual/Reference/Gcode_meta_commands`

7.4 Model Manipulation

Most of the Model Manipulation attacks work as expected. We attribute that to the G-Codes used in the attacks, as most of them are considered basic functionality. In fact, all printers behave similarly. Notable exceptions are the voids attack not working on the Ultimaker and the innovatiQ printers, as both do not support `M28`, and the innovatiQ printer not reacting to printing speed changes. Due to the similarity in the results, we decided to focus on a few examples from the Ender printer shown in Figure 8. Figure 8a depicts an unmodified reference print. Figures 8b and 8c depict over and under extrusion through the abuse of the volumetric extrusion mechanism (AC1). While the under extrusion is clear to see, the over extrusion is more subtle but still noticeable in direct comparison to the reference image. During testing, we noticed that Prusa has a range of G-Codes for checking parameters. This includes, for example, `M862.1` for checking if the nozzle diameter for which it was sliced is the same as the one reported by the printer. Note that this only prohibits accidental mismatches, as an attacker can reset the value themselves. The anomaly attacks on the `X/Y` and `Z` axes are shown in Figure 8d and 8e, respectively. Again, the Surface Anomaly is very obvious, while the Layer Height Anomaly can be noticed when focusing on the larger gaps in the layers when looking at the corners of the print.

None of the printers is vulnerable to attacks using `M143` for setting the maximum temperature. This is unsurprising, as the G-Code is only documented for the RepRap firmware. Setting the temperature through more common G-Codes (AC2) works as expected.



(a) Reference (b) Over Extrusion (c) Under Extrusion (d) Surface Anomaly (e) Layer Anomaly

Figure 8: Examples of successful Model Manipulation attacks on the Ender printer.

8 Conclusions and Future Work

The inherent problem of mixing control and data channels in printer systems highlights a significant security risk, as it creates a scenario where both printing tasks and configuration commands share the same access privileges. This is problematic because it allows potential security breaches. Historically, similar issues have been observed in traditional printer setups. For instance, in 2017, Müller et al. [41] have shown that improper separation of control and data channels can lead to unauthorized changes in printer configurations, which could be exploited to bypass security measures or to gain unintended access to sensitive data.

Notable documents from organizations such as the International Organization for Standardization (ISO) [26], the National Institute of Standards and Technology (NIST) [31], and the German Institute for Normalization (DIN) [15] provide guidelines on the use and structure of G-Codes. But, as these documents target Subtractive Manufacturing systems, they do not cover many of the G-Codes used in 3D printing today. Thus, many vendors implement proprietary G-Codes beyond the scope of the specification, sometimes leading to variations that are poorly documented or, in some cases, not documented at all. Projects like the RepRap wiki [49] try to create a comprehensive overview of the various G-Codes and aim to provide clarity and standardization. However, maintaining up-to-date documentation is an ongoing challenge, given the rapid pace of technological advancements and the proprietary nature of many implementations. As a result, the accuracy and completeness of these resources cannot always be guaranteed.

To handle the problems described in this paper, we discuss short-term solutions and outline long-term improvements as part of our future work.

8.1 Short-Term Solutions

To mitigate the security risks associated with mixed control and data channels in 3D printers, we propose two short-term countermeasures besides disabling access to the printer. While these countermeasures do not resolve the fundamental issue of separating control and data channels, they provide immediate, short-term solutions that limit the attack surface and enhance security.

Disabling Less-used Malicious G-Codes. Through our analysis of used G-Codes, we discovered that 162 of the 278 potentially malicious G-Codes are not used in any G-Code file on Thingiverse, and 210 are not used by slicers. Given their lack of relevance in many print jobs, these G-Codes could be disabled to reduce the potential attack surface. This countermeasure requires that firmware vendors provide a patch removing, disabling, or increasing the authorization level regarding dangerous G-Codes.

Overwrite G-Codes. In situations where it is not feasible to disable specific G-Codes, resetting the device before each print job serves as an effective countermeasure. By doing so, any long-living malicious settings are overwritten or reset. This reduces the risk of attacks, especially for shared 3D printers in environments like Makerspaces or online services.

This countermeasure requires actions by both the firmware and the operators providing the 3D printer. The firmware could support G-Codes for resetting all configurations, so an operator can execute the command after every print job.

G-Code Scanner. We implemented an open-source Python-based tool that addresses countermeasures against our attacks in two distinct ways. Firstly, the tool can analyze G-Code files before they are printed. This generates a report that identifies potentially dangerous G-Codes and discusses their significance regarding security. Secondly, it can scan 3D printers for supported G-Codes and output the same security report for the supported G-Codes. We implemented three modes (full, custom, and known) that define different depths of analysis.

8.2 Long-Term Improvements

We aim to highlight the primary reasons why the security of G-Codes needs improvements.

Clear Standardization. At first glance, many of the dangerous G-Codes could be disabled by the firmware, leading to the mitigation of many attacks. It should be considered that this countermeasure leads to compatibility issues with those slicers that may keep using forbidden G-Codes. In addition, control software such as Octoprint [23] relies on G-Codes to observe and adjust the printing process. These dependencies between existing software components involved in the printing process lead to complications regarding the implementation of countermeasures.

Due to the problems regarding the standardization of G-Codes and their handling, it is necessary to establish a clear guideline that all vendors implement. At this time, we do not believe that the requirements for such a standardization have been met.

Security Best Practices. From our perspective as security experts, without a standardized framework, there is no structured approach for disclosures of security vulnerabilities. It is unclear where the responsibility lies—whether with the firmware developers, the vendors, the slicers, or the G-Code documentation. This ambiguity in responsibility is confusing and makes addressing security issues difficult.

There are no established security best practices to guide vendors, firmware developers, or administrators. The awareness of the associated risks appears minimal, and there is no clarity on who is responsible for addressing and fixing these security vulnerabilities.

The Ultimaker is the only 3D printer that provides a higher level of security. The implementation of security standards such as ISO 27001 [28] and ISA/IEC 62443 [25] while adopting "the principle of least privilege" [58] prevents multiple attacks or requires a more powerful attacker. Nevertheless, the standards address basic security principles and do not explicitly target 3D printers and G-Codes. Thus, an interpretation gap for the firmware developers still exists. Considering the higher level of security that can be achieved, we encourage efforts to standardize security practices for 3D printers.

Modification Detection. Disabling certain G-Codes, particularly those frequently used in the printing process like `G1`, is impossible. Therefore, alternative countermeasures are essential, specifically in detecting model manipulations. Previous research has made significant contributions in this area. For example, Belikovetsky et al. [9] utilized audio signals emitted from a 3D printer to create a digital "signature" for process verification, while Gao et al. [20] proposed a monitoring approach to detect attacks on 3D models. Additionally, Blocklove [11] developed an FPGA-based intermediary to detect malicious changes to the G-Code.

Our work contributes to this field by expanding the list of G-Codes that can be used for Model Manipulation, thereby enhancing the training datasets for such detection tools. Future research should explore how our attacker models can be countered by detecting manipulations both before and during the printing process.

Summary. We suggest adopting standardization procedures and documents regarding the discussed security aspects. On the one side, the security awareness of potential G-Codes should be increased while the development of Secure Best Current Practices (SBCPs) encourages community involvement to contribute to and refine these standards. On the other side, future research should focus on the detection of malicious G-Codes which cannot be disabled.

Open Science

We provide all possible artifacts of our security analysis at `https://doi.org/10.5281/zenodo.14719309`. This includes

- the documentation of all G-Codes and the tools to extract the information,
- the tools used to obtain the usage data,
- source code that was used to generate a selection of test cases,
- detailed testing protocols that include information about the printer, firmware, notes, and photos of the printed objects, and
- a tool that users can run to determine which G-Codes are supported by their printer and if a G-Code file contains potentially malicious G-Codes.

Ethical Considerations

We are currently in the process of disclosing the immediate problems we found to the respective entities (i.e., firmware and hardware vendors). The group of 3D printer users is a large and diverse mixture of use cases, risk, and impacts coming from both individuals and companies. It is impossible to reach all of them prior to publication. We use a two-fold approach to mitigate this problem. First, we disclose to a national CERT, which can help reach the right people and companies. Additionally, we hope that the involvement of the CERT can help to improve current processes. Second, we also hope to use this paper as a communication tool to reach as many people as possible. The knowledge of the vulnerabilities allows people to protect themselves against attackers that might already exploit them. In many cases, the risk of our findings can be dramatically reduced by limiting access to the printer itself or to other defense-in-depth methods. Where that is not possible, we outlined alternative short-term solutions for the users to protect themselves (see Section 8.1). This two-fold approach allows both the companies and the users to decide for themselves how to proceed with the knowledge of these vulnerabilities, but also minimizes the risks as far as possible.

The tested 3D printers are hardware owned by us and tested without the possibility of damage to the hardware. Physical safety was considered while working with the printers. This includes an emergency power switch for all motors, safe operating distances, and cooldown periods for all heated elements.

Acknowledgements

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Germany's Excellence Strategy - EXC 2092 CASA–390781972 and by the research project "North-Rhine Westphalian Experts in Research on Digitalization (NERD II)", sponsored by the state of North Rhine-Westfalia – NERD II 005-2201-0014.

References

[1] 3Dnatives. How 3D Printing Allows for Customizable Consumer Goods, February 2023. URL: https://www.3dnatives.com/en/consumer-goods-3d-printing-08022023/.

[2] 3MF Consortium. 3D Manufacturing Format — Core Specification & Reference Guide v1.2.3. Technical report, 3MF Consortium, April 2024. URL: https://github.com/3MFConsortium/spec_core.

[3] Muhammad Ahsan, Muhammad Haris Rais, and Irfan Ahmed. SOK: Side Channel Monitoring for Additive Manufacturing - Bridging Cybersecurity and Quality Assurance Communities. In *2023 IEEE 8th European Symposium on Security and Privacy (EuroS&P)*, EuroS&P, pages 1160–1178, July 2023. doi:10.1109/EuroSP57164.2023.00071.

[4] Mohammad Abdullah Al Faruque, Sujit Rokka Chhetri, Arquimedes Canedo, and Jiang Wan. Acoustic side-channel attacks on additive manufacturing systems. In *2016 ACM/IEEE 7th international conference on cyber-physical systems*, ICCPS, pages 1–10, April 2016. doi:10.1109/ICCPS.2016.7479068.

[5] AMFG. Application Spotlight: 3D-Printed Rockets and the Future of Spacecraft Manufacturing, August 2019. URL: https://amfg.ai/2019/08/28/application-spotlight-3d-printed-rockets-and-the-future-of-spacecraft-manufacturing/.

[6] ANSI/EIA. RS-274-D Interchangeable variable block data format for positioning, contouring, and contouring/positioning numerically controlled machines, 1979.

[7] Caroline R. Arms, Carl Fleischhauer, Kate Murray, Marcus Nappier, and Liz Holdzkom. STL (STereoLithography) File Format Family, April 2024. URL: https://www.loc.gov/preservation/digital/formats/fdd/fdd000504.shtml.

[8] Caleb Beckwith, Harsh Sankar Naicker, Svara Mehta, Viba R. Udupa, Nghia Tri Nim, Varun Gadre, H. Pearce, Gary Mac, and Nikhil Gupta. Needle in a Haystack: Detecting Subtle Malicious Edits to Additive Manufacturing G-code Files. *IEEE Embedded Systems Letters*, 2021. doi:10.1109/LES.2021.3129108.

[9] Sofia Belikovetsky, Yosef A. Solewicz, Mark Yampolskiy, Jinghui Toh, and Yuval Elovici. Digital Audio Signature for 3D Printing Integrity. *IEEE Transactions on Information Forensics and Security*, 14(5):1127–1141, May 2019. Conference Name: IEEE Transactions on Information Forensics and Security. doi:10.1109/TIFS.2018.2851584.

[10] Sofia Belikovetsky, Mark Yampolskiy, Jinghui Toh, Jacob Gatlin, and Yuval Elovici. dr0wned – cyber-physical attack with additive manufacturing. In *11th USENIX workshop on offensive technologies*, WOOT 17, Vancouver, BC, August 2017. USENIX Association. URL: https://www.usenix.org/conference/woot17/workshop-program/presentation/belikovetsky.

[11] Jason Blocklove, Md Raz, Prithwish Basu Roy, Hammond Pearce, Prashanth Krishnamurthy, Farshad Khorrami, and Ramesh Karri. OffRAMPS: An FPGA-based

Intermediary for Analysis and Modification of Additive Manufacturing Control Systems. In *Proceedings of the 54th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*, April 2024. arXiv:2404.15446 [cs, eess]. doi:10.48550/arXiv.2404.15446.

[12] Sujit Rokka Chhetri, Anomadarshi Barua, Sina Faezi, Francesco Regazzoni, Arquimedes Canedo, and Mohammad Abdullah Al Faruque. Tool of Spies: Leaking your IP by Altering the 3D Printer Compiler. *IEEE Transactions on Dependable and Secure Computing*, 18(2):667–678, March 2021. Conference Name: IEEE Transactions on Dependable and Secure Computing. doi:10.1109/TDSC.2019.2923215.

[13] Creative-Tools.com. #3DBenchy, 2022. CC BY-ND 4.0. URL: https://www.3dbenchy.com/.

[14] daid. Inside the Ultimaker 3 - Day 1 - GCode, October 2016. URL: https://community.ultimaker.com/topic/15555-inside-the-ultimaker-3-day-1-gcode/.

[15] DIN. 66025-1. Programmaufbau für numerisch gesteuerte Arbeitsmaschinen; Allgemeines, January 1983.

[16] Quang Do, Ben Martini, and Kim-Kwang Raymond Choo. A Data Exfiltration and Remote Exploitation Attack on Consumer 3D Printers. *IEEE Transactions on Information Forensics and Security*, 11(10):2174–2186, October 2016. Conference Name: IEEE Transactions on Information Forensics and Security. doi:10.1109/TIFS.2016.2578285.

[17] Duet3D. Duet Web Control Manual, October 2024. URL: https://docs.duet3d.com/User_manual/Reference/Duet_Web_Control_Manual.

[18] FlashForge. Gcode Protocol v1.04 (Partial). URL: https://github.com/minsk-hackerspace/slic3r-configs/blob/eede7b3391bf2b7c776635c29708ecf16d77f8da/docs/Flashforge%20Gcode%20protocol(open).md.

[19] Fortune Business Insights. 3D Printing Market Size, Growth, Share | Global Report [2032], August 2024. URL: https://www.fortunebusinessinsights.com/industry-reports/3d-printing-market-101902.

[20] Yang Gao, Borui Li, Wei Wang, Wenyao Xu, Chi Zhou, and Zhanpeng Jin. Watching and Safeguarding Your 3D Printer: Online Process Monitoring Against Cyber-Physical Attacks. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 2(3):108:1–108:27, September 2018. doi:10.1145/3264918.

[21] Jacob Gatlin, Sofia Belikovetsky, Yuval Elovici, Anthony Skjellum, Joshua Lubell, Paul Witherell, and Mark Yampolskiy. Encryption is Futile: Reconstructing 3D-Printed Models Using the Power Side-Channel. In *24th International Symposium on Research in Attacks, Intrusions and Defenses*, RAID '21, pages 135–147. Association for Computing Machinery, October 2021. doi:10.1145/3471621.3471850.

[22] Avesta Hojjati, Anku Adhikari, Katarina Struckmann, Edward Chou, Thi Ngoc Tho Nguyen, Kushagra Madan, Marianne S. Winslett, Carl A. Gunter, and William P. King. Leave your phone at the door: Side channels that reveal factory floor secrets. In *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, CCS '16, pages 883–894. Association for Computing Machinery, 2016. doi:10.1145/2976749.2978323.

[23] Gina Häußge. OctoPrint, 2024. URL: https://octoprint.org/.

[24] ideaMaker and JZ. Omit RaiseTouch Specified Gcode. Section: article:section ideaMaker Library: Find the best ideaMaker profile for your 3D printer. URL: https://www.ideamaker.io/dictionaryDetail.html?name=Omit%20RaiseTouch%20Specified%20Gcode.

[25] ISA/IEC. 62443 Series. Security for industrial automation and control systems, 2024. URL: https://www.isa.org/standards-and-publications/isa-standards/isa-iec-62443-series-of-standards.

[26] ISO. 6983-1. Automation systems and integration. Numerical control of machines. Program format and definitions of address words. Part 1: Data format for positioning, line motion and contouring control systems, December 2009. URL: https://www.iso.org/standard/34608.html.

[27] ISO/ASTM. 52915. Specification for additive manufacturing file format (AMF) Version 1.2, March 2020. URL: https://www.iso.org/standard/74640.html.

[28] ISO/IEC. 27001. Information security, cybersecurity and privacy protection — Information security management systems — Requirements, October 2022. URL: https://www.iso.org/standard/27001.

[29] John M. Evans, Jr., et. al. Standards for Computer Aided Manufacturing. Final Report NBSIR 76-1094 (R), National Bureau of Standards, Washington, D. C., June

1977. URL: https://nvlpubs.nist.gov/nistpubs/Legacy/IR/nbsir76-1183.pdf.

[30] Klipper. G-Codes, January 2024. URL: https://github.com/Klipper3d/klipper/blob/master/docs/G-Codes.md.

[31] Thomas Kramer, Frederick Proctor, and Elena Messina. NISTIR 6556. The NIST RS274NGC Interpreter - Version 3, August 2000. URL: https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=823374.

[32] Scott Lahteine. Comment on Issue #11934 of MarlinFirmware/Marlin, September 2018. URL: https://github.com/MarlinFirmware/Marlin/issues/11934#issuecomment-425274674.

[33] MachMotion. Mach4 Advanced M-Code Reference. URL: https://support.machmotion.com/books/software/page/mach4-advanced-m-code-reference.

[34] Marlin. G-code Index, January 2024. URL: https://marlinfw.org/meta/gcode/.

[35] Matthew McCormack, Sanjay Chandrasekaran, Guyue Liu, Tianlong Yu, Sandra DeVincent Wolf, and Vyas Sekar. Security Analysis of Networked 3D Printers. In *2020 IEEE Security and Privacy Workshops (SPW)*, pages 118–125, May 2020. doi:10.1109/SPW50608.2020.00035.

[36] MK4duo. GCodes, May 2020. URL: https://github.com/MKFirmware/MK4duo/blob/master/Documentation/GCodes.md.

[37] Abdul Aleem Mohammed, Mohammed S. Algahtani, Mohammad Zaki Ahmad, Javed Ahmad, and Sabna Kotta. 3D Printing in medicine: Technology overview and drug delivery applications. *Annals of 3D Printed Medicine*, 4:100037, December 2021. doi:10.1016/j.stlm.2021.100037.

[38] Samuel Bennett Moore, Phillip Armstrong, Todd McDonald, and Mark Yampolskiy. Vulnerability analysis of desktop 3D printer software. In *2016 Resilience Week (RWS)*, pages 46–51. IEEE, August 2016. doi:10.1109/RWEEK.2016.7573305.

[39] Samuel Bennett Moore, William Bradley Glisson, and Mark Yampolskiy. Implications of Malicious 3D Printer Firmware. In *Hawaii International Conference on System Sciences 2017 (HICSS-50)*, January 2017. doi:10125/41899.

[40] Mordor Intelligence Research & Advisory. 3D Printing Market Size & Share Analysis - Growth Trends & Forecasts (2024 - 2029), March 2024. URL: https://www.mordorintelligence.com/industry-reports/3d-printing-market.

[41] Jens Müller, Vladislav Mladenov, Juraj Somorovsky, and Jörg Schwenk. SoK: Exploiting Network Printers. In *2017 IEEE Symposium on Security and Privacy (SP)*, pages 213–230, May 2017. ISSN: 2375-1207. doi:10.1109/SP.2017.47.

[42] Allie Nawrat. 3D printing in the medical field: four major applications revolutionising the industry, August 2018. URL: https://www.medicaldevice-network.com/features/3d-printing-in-the-medical-field-applications/.

[43] H. Pearce, K. Yanamandra, Nikhil Gupta, and R. Karri. FLAW3D: A Trojan-based Cyber Attack on the Physical Outcomes of Additive Manufacturing. *IEEE/ASME Transactions on Mechatronics*, 2022. doi:10.1109/tmech.2022.3179713.

[44] Melissa Pistilli. 3D Printing Stocks: 9 Biggest Companies, August 2023. URL: https://investingnews.com/daily/tech-investing/emerging-tech-investing/top-3d-printing-companies/.

[45] Prusa. Buddy firmware-specific G-code commands, 2023. URL: https://help.prusa3d.com/article/buddy-firmware-specific-g-code-commands_633112.

[46] Muhammad Haris Rais, Muhammad Ahsan, and Irfan Ahmed. SOK: 3D Printer Firmware Attacks on Fused Filament Fabrication. In *Proceedings of the 18th USENIX WOOT Conference on Offensive Technologies*, pages 263–282, 2024. URL: https://www.usenix.org/conference/woot24/presentation/rais.

[47] Repetier. Implemented GCodes, September 2019. URL: https://github.com/repetier/Repetier-Firmware/blob/master/src/ArduinoAVR/Repetier/Repetier.ino.

[48] RepRap. Fused filament fabrication, February 2019. URL: https://reprap.org/wiki/Fused_filament_fabrication.

[49] RepRap. G-code, June 2024. URL: https://reprap.org/wiki/G-code.

[50] Jost Rossel. Usage Statistics of 3D Printing File Formats, 2022. URL: https://upb-syssec.github.io/blog/2022/3d-printing-file-format-usage/.

[51] Jost Rossel, Vladislav Mladenov, and Juraj Somorovsky. Security Analysis of the 3MF Data Format. In *Proceedings of the 26th International Symposium on Research in Attacks, Intrusions and Defenses*, RAID '23, pages 179–194. Association for Computing Machinery, October 2023. doi:10.1145/3607199.3607216.

[52] Sculpteo. Which 3D printing technologies do you use?, April 2021. URL: https://www.statista.com/statistics/560304/worldwide-survey-3d-printing-top-technologies/.

[53] Chen Song, Feng Lin, Zhongjie Ba, Kui Ren, Chi Zhou, and Wenyao Xu. 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 '16, pages 895–907. Association for Computing Machinery, 2016. Number of pages: 13 Place: Vienna, Austria. doi:10.1145/2976749.2978300.

[54] Logan D. Sturm, Christopher B. Williams, Jamie A. Camelio, Jules White, and Robert Parker. Cyber-physical vulnerabilities in additive manufacturing systems: A case study attack on the .STL file with human subjects. *Journal of Manufacturing Systems*, 44:154–164, July 2017. doi:10.1016/j.jmsy.2017.05.007.

[55] Team Xometry. G-Code: Definition, Function, and Different Types, October 2022. URL: https://www.xometry.com/resources/machining/what-is-g-code/.

[56] Nektarios Georgios Tsoutsos, Homer Gamil, and Michail Maniatakos. Secure 3D Printing: Reconstructing and Validating Solid Geometries using Toolpath Reverse Engineering. In *Proceedings of the 3rd ACM Workshop on Cyber-Physical System Security*, CPSS '17, pages 15–20. Association for Computing Machinery, April 2017. doi:10.1145/3055186.3055198.

[57] Hamilton Turner, Jules White, Jaime A. Camelio, Christopher Williams, Brandon Amos, and Robert Parker. Bad parts: Are our manufacturing systems at risk of silent cyberattacks? *IEEE Security and Privacy*, 13(3):40–47, May 2015. doi:10.1109/MSP.2015.60.

[58] Ultimaker. UltiMaker security, October 2024. URL: https://support.ultimaker.com/s/article/1667411337398.

[59] United States Patent and Trademark Office. Trademark 74133656 (FDM), 1991. URL: https://tsdr.uspto.gov/#caseNumber=74133656&caseType=SERIAL_NO&searchType=statusSearch.

[60] Wavefront Technologies. The Advanced Visualizer — Appendix B1. Object Files (.obj). URL: https://www.loc.gov/preservation/digital/formats/fdd/fdd000507.shtml.

[61] Xun Xu. *Integrating Advanced Computer-Aided Design, Manufacturing, and Numerical Control: Principles and Implementations*. Information Science Reference (an imprint of IGI Global), January 2009. doi:10.5555/1524218.

[62] Mark Yampolskiy, Wayne King, Gregory Pope, Sofia Belikovetsky, and Yuval Elovici. Evaluation of Additive and Subtractive Manufacturing from the Security Perspective. In Mason Rice and Sujeet Shenoi, editors, *Critical Infrastructure Protection XI*, pages 23–44. Springer International Publishing, 2017. doi:10.1007/978-3-319-70395-4_2.

[63] Mark Yampolskiy, Wayne E. King, Jacob Gatlin, Sofia Belikovetsky, Adam Brown, Anthony Skjellum, and Yuval Elovici. Security of additive manufacturing: Attack taxonomy and survey. *Additive Manufacturing*, 21:431–457, May 2018. doi:10.1016/j.addma.2018.03.015.

[64] Wei Yang, Jialei Chen, Chuck Zhang, and Kamran Paynabar. Online detection of cyber-incidents in additive manufacturing systems via analyzing multimedia signals. *Quality and Reliability Engineering International*, 38(3):1340–1356, 2022. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/qre.2953. doi:10.1002/qre.2953.

[65] Ronan Ye. What is G-code: Definition, Function, Types & Uses, July 2023. URL: https://www.3erp.com/blog/g-code/.

[66] Zhiyuan Yu, Yuanhaur Chang, Shixuan Zhai, Nicholas Deily, Tao Ju, XiaoFeng Wang, Uday Jammalamadaka, and Ning Zhang. XCheck: Verifying Integrity of 3D Printed Patient-Specific Devices via Computing Tomography. In *Proceedings of the 32nd USENIX Security Symposium*, USENIX Security 23, pages 2815–2832, 2023. URL: https://www.usenix.org/conference/usenixsecurity23/presentation/yu-zhiyuan-xcheck.

[67] Steven Eric Zeltmann, Nikhil Gupta, Nektarios Georgios Tsoutsos, Michail Maniatakos, Jeyavijayan Rajendran, and Ramesh Karri. Manufacturing and Security Challenges in 3D Printing. *JOM*, 68(7):1872–1881, July 2016. doi:10.1007/s11837-016-1937-7.

A Appendix

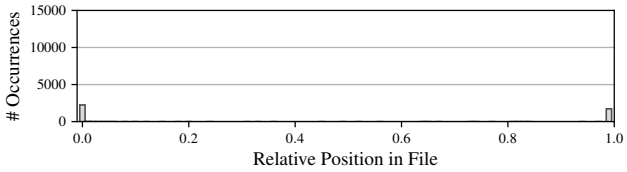


Figure A.9: Histogram of the positions where `M221` is used.

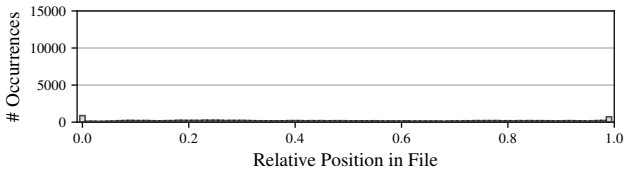


Figure A.10: Histogram of the positions where `M220` is used.

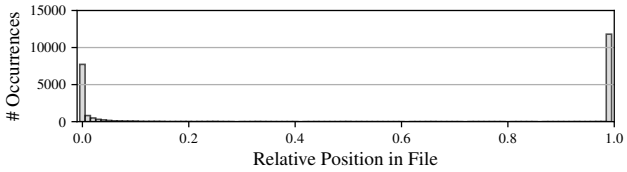


Figure A.11: Histogram of the positions where `M140` is used.

Table A.4: Slicers and Used G-Codes

Slicer	# Default Configurations	# Unique G-Codes
PrusaSlicer 2.7.2	213	81
UltiMaker Cura 5.3.1	126	78
Creality Slicer 4.8.2	70	22
FlashPrint 5.8.3	26	29
ideaMaker 4.3.3	13	21
Slic3r 1.3.0	8	19
Repetier-Host 2.3.2	4	14
Tinkerine Suite 3.0	4	14
Combined	464	129

The number of default configurations available in the slicer (equal to the amount of generated G-Code files) and how many unique G-Codes are present in these generated files.

Table A.5: All Documented G-Codes Categorized

Attack	G-Codes
Model Manipulation	
├ Toolpath Manipulation	G0, G1, G2, G3, G5, G6, G10, G30, G92, G161, G162, G280, M18, M28, M84, M206, M425
│ ├ Voids	G0, G1, G2, G3, G5, G6, G30, M28
│ ├ Infill Anomaly	G0, G1, G2, G3, G5, G6, G30
│ ├ Surface Anomaly (X/Y Shift)	G0, G1, G2, G3, G5, G6, G10, G161, G162, G280, M18, M84, M425
│ ├ Layer Height Anomaly (Z Shift)	G0, G1, G2, G3, G5, G6, G10, G92, M18, M84, M206
│ ├ Layer Height	G0, G1, G2, G3, G5, G6
│ └ Print Angle	G0, G1, G2, G3, G5, G6
├ Faulty Extrusion	
│ ├ Under Extrusion	G0, G1, G2, G3, G5, G6, G10, G201, G280, M18, M82, M83, M84, M200, M221, M404, M592, M702, M703, M900, M1701
│ ├ Over Extrusion	G0, G1, G2, G3, G5, G6, G10, G201, G280, M18, M82, M83, M84, M200, M221, M404, M592, M703, M900
│ ├ Material Relocation	G0, G1, G2, G3, G5, G6
│ ├ Filament Retraction	G0, G1, G2, G3, G5, G6, G10, G11, G22, G23, G280, M101, M103, M207, M208, M209, M227, M228, M229
│ └ Printing Speed	G0, G1, G2, G3, G5, G6, G93, G94, M201, M201.1, M202, M203, M204, M205, M220, M222, M223, M566
└ Temperature Changes	
├ Bed Temperature	M108, M140, M144, M149, M170, M190, M230, M301, M304, M305, M307, M309, M568, M570, M912
├ Nozzle Temperature	G280, M8, M9, M13, M104, M108, M109, M143, M149, M170, M230, M301, M302, M305, M307, M309, M568, M570, M912
└ Fan Speed	M106, M107, M245, M246, M460, M652, M710
Information Disclosure	
├ Intellectual Property Theft	M111, M154, M928
└ Metadata Leakage	D2, D3, D4, D5, D6, D7, D8, D9, D20, D21, D80, D81, D106, G, G31, G32, G64, G75, G81, G203, M, M16, M20, M27, M31, M33, M36, M39, M44, M46, M73, M78, M100, M105, M111, M113, M114, M115, M117, M119, M122, M123, M136, M155, M205, M302, M307, M309, M310, M334, M360, M407, M408, M409, M430, M450, M493, M501, M503, M544, M553, M554, M573, M590, M701, M710, M860, M861, M862, M862.1, M862.2, M862.3, M862.4, M862.5, M862.6, M863, M864, M865, M866, M867, M868, M869, M900, M906, M920, M922, M929, M951, M956, M993
Denial of Service	
├ Interrupt Printing	
│ ├ Infinite Loop	D-1, M26, M41, M808, M810, M811, M812, M813, M814, M815, M816, M817, M818, M819
│ ├ Delay Commands	G4, G12, G26, G27, G28, G29, G30, G32, G33, G34, G35, G38, G38.2, G38.3, G38.4, G38.5, G42, G61, G65, G82, G132, G133, G161, G162, G163, G204, G280, G425, M0, M1, M25, M45, M47, M112, M116, M125, M226, M291, M361, M362, M363, M364, M371, M372, M400, M401, M510, M577, M585, M601, M675, M700, M701, M702, M705, M706
│ ├ Ignore Commands / Stop Print	G93, G204, M16, M22, M28, M30, M37, M40, M81, M92, M110, M112, M124, M410, M412, M452, M472, M486, M524, M544, M563, M588, M599, M603, M604, M709, M906
│ └ Destroy Model / Make Unusable	G0, G1, G2, G3, G5, G6, G17, G18, G19, G20, G21, G26, G30, G33, G34, G35, G61, G65, G68, G82, G90, G91, G92, G92.x, G93, G130, G201, G280, M82, M83, M206, M214, M365, M401, M406, M428, M452, M579, M606, M913
└ Disable Access / Bricking	
├ Software	D0, D2, M16, M35, M510, M540, M551, M552, M553, M554, M559, M560, M575, M586, M586.4, M587, M588, M599
└ Hardware	G0, G1, G2, G3, G5, G6, G10, G82, M302, M564, M906, M913