Chapter 6

MANIPULATION OF G-CODE TOOLPATH FILES IN 3D PRINTERS: ATTACKS AND MITIGATIONS

Elizabeth Kurkowski, Alyxandra Van Stockum, Joel Dawson, Curtis Taylor, Tricia Schulz and Sujeet Shenoi

Abstract Additive manufacturing or 3D printing is commonly used to create mission-critical parts in the critical infrastructure. This research focuses on threats that target the key slicing step of additive manufacturing, when design files that model part geometry are converted to G-code toolpath files that convey instructions for printing parts layer by layer. The research leverages a hitherto unknown slicing software vulnerability where G-code corresponding to part slices is stored as plaintext ASCII characters in heap memory during execution. The vulnerability was discovered in two open-source, full-featured slicing software suites that support many 3D printers.

Experiments with a toolkit developed to target slicing software in real time demonstrate that the attacks are surreptitious and fine-grained. Two attacks, temperature modification and infill exclusion, performed against G-code generated for fused filament fabrication printers demonstrate the ability to sabotage printed parts as well as print environments. Although the vulnerability can be mitigated using strong authentication and access controls along with G-code obfuscation, the ability to automate surreptitious, fine-grained attacks that degrade printed parts in ways that are imperceptible to the human eye and undetectible by non-destructive testing methods is a serious concern.

Keywords: Additive manufacturing, fused filament fabrication, G-code attacks

1. Introduction

Additive manufacturing (AM) or 3D printing is the process of depositing layers of material to create 3D objects. Additive manufacturing is a

J. Staggs and S. Shenoi (Eds.): Critical Infrastructure Protection XVI, IFIP AICT 666, pp. 155–174, 2022.
https://doi.org/10.1007/978-3-031-20137-0_6

rapidly-growing segment of the manufacturing sector; its market value increased by 21.2% to $11.867 billion in 2019 alone [11].

Additive manufacturing is a competitive alternative to traditional subtractive manufacturing with many economic and environmental advantages. It enables complex internal structures to be created during a single print run due to the layer-by-layer part printing process. To obtain similar results, traditional manufacturing often requires the creation and assembly of multiple individual parts. Additively-manufactured parts often have improved mechanical properties due to high resolution control over internal structures [6]. Additive manufacturing enables rapid prototyping due to quick design-to-product times as well as on-demand low volume or high-volume production. Also, additive manufacturing has less material wastage than subtractive manufacturing.

Additive manufacturing is heavily utilized in many critical infrastructure sectors, including energy, healthcare, transportation and defense [3–5, 8, 16]. However, a 2021 cyber security audit of defense additive manufacturing systems by the Inspector General of the U.S. Department of Defense [7] determined that all the reviewed sites did not consistently manage or secure their systems to prevent unauthorized changes and ensure the integrity of design data.

This research focuses on threats targeting the key slicing step of additive manufacturing. During this step, part design files that model part geometry are converted to G-code toolpath files that cover printer actions and parameters such as extruder movements in each print layer, print speed, melt block temperature, material extrusion amount and fan speed. G-code toolpath files may be targeted by deleting sections of code (data destruction attacks) or by modifying the code (data integrity attacks), potentially sabotaging printed parts and print environments.

This research leverages a hitherto unknown vulnerability that G-code corresponding to part slices is stored as plaintext ASCII characters in heap memory during execution. The vulnerability was discovered in two open-source, full-featured slicing software suites that support many 3D printers. Exploiting the vulnerability requires root access (full code execution privileges) to the controller machine that executes the slicing software. The attacks access the runtime process image, scan the image memory, extract the plaintext G-code, perform various modifications to the G-code and write the code back to image memory.

Experiments using a toolkit developed to target slicing software in real time demonstrate that the attacks are surreptitious (difficult to detect) and fine-grained (able to target specific layers of printed parts). The two experimental attacks, temperature modification and infill exclusion, performed against G-code generated for fused filament fabrication

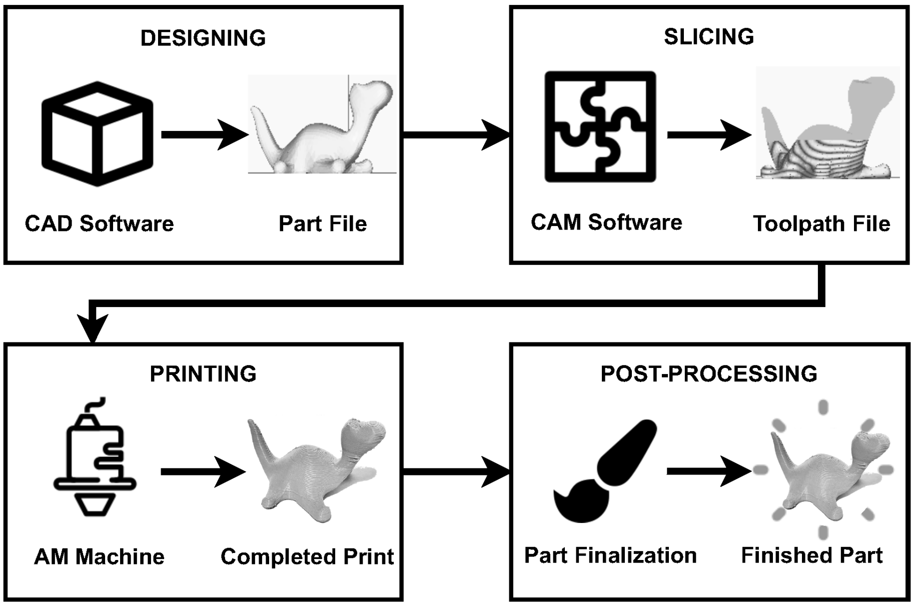

Figure 1. Additive manufacturing process chain.

printers demonstrate the ability to sabotage printed parts and print environments. The temperature modification attacks, which reduced the extruder head temperature from 198°C to 190°C while printing just seven of the 530 total layers, resulted in a 14% drop in the average tensile strength. The infill exclusion attacks, which excluded infill from five and 25 layers of the 127 total layers of printed parts, reduced the average compressive strengths by 10.6% and 19.9%, respectively. The ability to automate surreptitious, fine-grained attacks that significantly degrade printed parts in ways that are imperceptible to the human eye and undetectible by nondestructive testing methods demands systematic efforts at securing additive manufacturing systems from cyber threats.

2. Background and Related Work

This section describes the additive manufacturing process chain and research focused on attacking the process chain.

2.1 Additive Manufacturing Process Chain

The additive manufacturing process chain has four steps: designing, slicing, printing and post-processing. Figure 1 shows the steps in the process chain.

The designing step uses computer-aided design (CAD) software to create a part design model that is specified as a part file. The popular stereolithography (STL) part file format uses triangles to model the geometry of a part [10]. The more advanced 3MF file format captures the geometry of the desired part as well as its properties such as materials and colors [1].

During the slicing step, a part file is imported by computer-aided manufacturing (CAM) software. In additive manufacturing, this software is referred to as a slicer because it cuts the 3D-part model into 2D layers for printing.

The slicer enables a user to specify various print options before generating a toolpath file for a printer. Print options of interest include adding support material, specifying infill properties and characteristics such as density and pattern, and selecting the print speed and orientation. Support material is necessary when a print orientation causes part overhangs. Other print settings such as infill characteristics and print speed can affect the strength and other mechanical properties of the printed parts. All these features are eventually encoded in a toolpath file for printing. The toolpath commands, commonly called G-code, cover printer actions and parameters such as extruder movements, print speed, melt block temperature, material extrusion amount and fan speed.

During the printing step, a printer follows the instructions in a toolpath file to print a part. The printing time varies based on part size, geometry and material.

After a part is printed, one or more post-processing steps may be performed. In some cases, post-processing simply involves removing the support material and sanding rough edges. In other cases, especially for metal parts, annealing is performed to obtain the desired mechanical properties. The final post-processing step is quality control, which ensures that a printed part meets the design specifications.

2.2 Process Chain Attacks

Due to the relative youth of the field, security research on additive manufacturing systems is limited. Attacks on additive manufacturing fall into two main categories, theft of technical data and sabotage of parts or print environments [20]. While malicious attacks against 3D printers are rare, several potential attacks during the first three steps of the process chain have been theorized. Attacks during the last (fourth) post-processing step are possible. However, they are highly specific to the printing materials and parts. Therefore, they are rarely discussed in the literature. In any case, they are outside the scope of this research.

During the designing step, attacks typically target part design files. For example, Belikovetsky et al. [2] proposed a phishing attack to install a backdoor on the computer hosting the design software; the backdoor is leveraged to compromise the STL part design file and weaken the printed parts. Sturm et al. [18] demonstrated that malicious STL file modifications can cause printed parts to fail prematurely.

Zeltmann et al. [21] theorized several attacks during the slicing step. These include embedding defects and changing part orientations when the printing slices are created, sabotaging the parts created during the printing step.

Attacks targeting the printing step focus heavily on modifying 3D-printer firmware. Xiao [19] modified the firmware of a desktop RepRap Prusa printer to alter the temperature feedback loop, leading the printer to believe that the extruder temperature was double the real temperature.

Moore et al. [13] performed static and dynamic code analyses of Cura 3D, ReplicatorG, Repetier-Host and Marlin 3D-printer firmware that revealed several vulnerabilities in their code bases. These included buffer overflows and unencrypted host-printer communications. The security implications of these vulnerabilities include theft of technical data and part sabotage. Moore and colleagues also noted that weaknesses in the G-code structure provide opportunities for printed part manipulation. In subsequent work, Moore et al. [14] introduced malicious modifications to Marlin 3D-printer firmware. The modified firmware ignored incoming print commands, substituted malicious print commands in place of legitimate commands and manipulated extruder feed rates.

Pearce et al. [15] used a Trojan bootloader to infiltrate Marlin-compatible 3D printers and compromise the integrity of printed parts. The attack leveraged bootloader control over the initial printer firmware installation. The bootloader was able to scan for and modify byte patterns in the firmware, triggering G-code manipulations that reduced the extrusion rate and reordered print commands.

Rais et al. [17] developed novel attacks on fused filament fabrication printers. Their dynamic thermal and localized filament kinetic attacks were executed by a printer firmware rootkit that modified G-code toolpath instructions. The attacks had minimal footprints, but the damage to the printed parts was significant.

The research efforts described above and other research in additive manufacturing theorize potential attacks against design files and printer firmware, but few actually implement proof-of-concept attacks. In contrast, this research, which focuses on toolpath files created during the slicing step, allows for generalizable, surreptitious attacks that leverage weaknesses in slicing software. The research has also developed a tool for deploying the attacks on real 3D-printer process chains.

3. G-Code Toolpath File Attack Surface

This research targets G-code toolpath files by deleting sections of the G-code (data destruction attacks) or modifying the G-code (data integrity attacks). Both types of attacks change print layer information, sabotaging the printed parts.

An attack vector is a means for gaining access to a target, in this case, an entire G-code toolpath file or portions of its code. The attack surface of a G-code toolpath file is the collection of attack vectors that target the designing, slicing and printing steps of the process chain. Since the three steps are chained, an attack vector that provides access to a component or service during a preceding step provides indirect access to a G-code toolpath file in a later step. For example, gaining access to CAD software in the designing step enables malicious modifications to the part design file, which are encoded in the G-code toolpath file used during the slicing step.

The following attack vectors provide (direct or indirect) access to the G-code toolpath file/code during the designing, slicing and printing steps:

Designing Step Attack Vectors.

- **Access to CAD Software from Controller Computer:** An attacker can introduce malicious modifications to CAD software and/or its runtime process image that change part files unbeknownst to a user.

- **Access to CAD Software via Network:** An attacker can introduce malicious modifications to CAD software and/or its runtime process image that change part files unbeknownst to a user.

- **Access to CAD Software via Remote Software Update:** An attacker can introduce malicious modifications to a CAD software update that change part files unbeknownst to a user.

- **Access to Part Files from Controller Computer:** An attacker can introduce malicious modifications by manually editing part files unbeknownst to a user.

- **Access to Part Files via Network:** An attacker with remote access to a controller computer can introduce malicious modifications by manually editing part files unbeknownst to a user. If the part files are transferred to another controller computer for slicing, an attacker can assume a man-in-the-middle position to introduce malicious modifications to part files during transfer.

Slicing Step Attack Vectors.

- **Access to Slicing (CAM) Software from Controller Computer:** An attacker can introduce malicious modifications to slicing software and/or its runtime process image that change G-code toolpath files unbeknownst to a user. The attacks developed in this research can leverage this attack vector to target the runtime process image of slicing software.

- **Access to Slicing (CAM) Software via Network:** An attacker can introduce malicious modifications to slicing software and/or its runtime process image that change G-code toolpath files unbeknownst to a user. The attacks developed in this research can leverage this attack vector to target the runtime process image of slicing software.

- **Access to Slicing (CAM) Software via Remote Software Update:** An attacker can introduce malicious modifications to a CAM software update that change G-code toolpath files unbeknownst to a user. The attacks developed in this research can leverage this attack vector to target the runtime process image of slicing software.

- **Access to Toolpath Files from Controller Computer:** An attacker can introduce malicious modifications by manually editing G-code toolpath files unbeknownst to a user.

- **Access to Toolpath Files via Network:** An attacker with remote access to a controller computer can introduce malicious modifications by manually editing G-code toolpath files unbeknownst to a user. An attacker can also assume a man-in-the-middle position to introduce malicious modifications to G-code toolpath files during transfer to a printer.

Printing Step Attack Vectors.

- **Access to Toolpath Files via Network:** Networked printers enable users to remotely issue print commands and monitor the print status. An attacker with remote access can maliciously modify G-code toolpath commands unbeknownst to a user.

- **Access to Toolpath Files from Printer:** An attacker can introduce malicious modifications by manually editing G-code toolpath files unbeknownst to a user.

- **Access to Toolpath Files via Printer Firmware:** An attacker can introduce malicious modifications to printer firmware that change G-code toolpath files unbeknownst to a user.

4. G-Code Toolpath File Exploitation

This section describes a vulnerability discovered in slicer software from multiple vendors that can be exploited to attack G-code toolpath files during the slicing step. Exploiting the vulnerability requires access to the runtime process image of the slicing software from the controller computer or network.

4.1 Software Execution Vulnerability

The slicing software vulnerability is that G-code corresponding to part slices is stored in an unprotected manner as plaintext ASCII characters in heap memory during execution. The vulnerability was discovered in slicing software suites from two vendors. The software suites were selected because they are open-source, full-featured and support many 3D printers. For security reasons, details about the slicing software suites are obfuscated. The two slicing software suites are referred to as Alpha and Beta.

4.2 Software Execution Attack

The attacker requires root access (full code execution privileges) to the controller machine that executes the slicing software. The attack accesses the runtime process image, scans the image memory, extracts the plaintext G-code, modifies the G-code and writes it back to image memory.

Figure 2 shows the details of the slicing software execution attack. An unsuspecting user loads a part file, slices the part and previews the G-code toolpath data using the slicer graphical user interface. While the user is previewing the slice layers, the attack modifies the G-code in runtime process image memory. Since the memory modifications occur in the background, the graphical user interface continues to present the original unmodified slices.

After previewing the G-code, the unsuspecting user proceeds to save the G-code to a file. However, the modified G-code in heap memory is saved to the toolpath file instead of the original G-code.

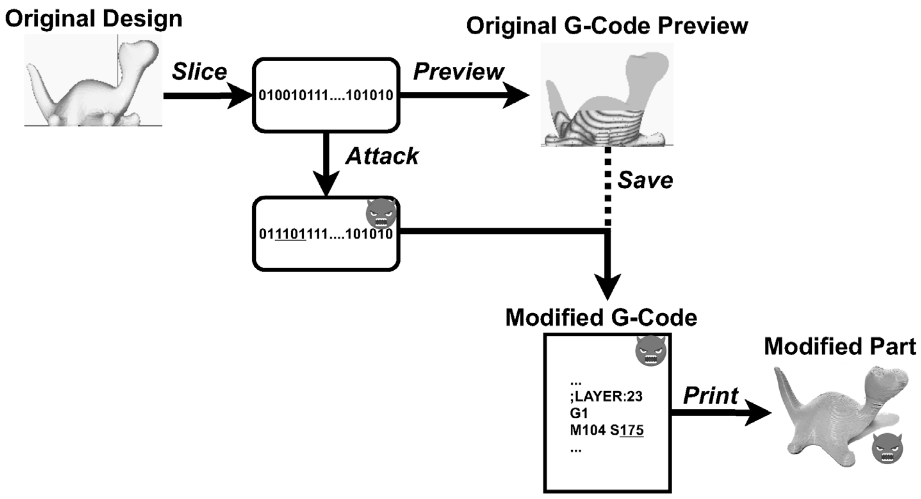

Figure 2. Slicing software execution attack.

A toolkit was created to launch slicing software execution attacks. The toolkit conducts the attack in two phases, layer identification and layer modification:

- **Layer Identification:** The first attack phase involves scanning the slicer process heap memory address range for plaintext G-code layers. The goal is to reconstruct the original G-code toolpath file to the extent possible.

  Each slicer has a slightly different format for header data that can be used to extract summary data about a print. Each slicer also has specific start and end delimiters for a toolpath layer that can be used to track G-code in heap memory. When a G-code layer is being processed, all the data associated with the layer is stored in consecutive pages in memory. However, successive layers are not necessarily stored contiguously. Therefore, the individual layers of G-code in memory have to be identified. Algorithm 1 specifies the layer identification procedure.

- **Layer Modification:** After the G-code is reconstructed from memory, modifications are made to the code. Since G-code has a standard format [9], it is straightforward to implement attacks given a complete G-code reconstruction. Layer modification involves parsing the reconstructed G-code, making the malicious G-code modifications and writing the modified G-code back to the same locations in memory.

Algorithm 1: Identify G-code layers in process heap memory.

```
Input: startAddress: Start address of slicer process heap memory
Output: layerTable: Hash table containing layer information
layerTable ← LayerStruct[]
page ←initialPage(startAddress)
while page ≠ End of Memory do
    page ← getNextPage()
    if startConditionExists then
        layer ← new LayerStruct
        layer.id ←parseID()
        layer.start ← getStartLocation()
        while !endConditionExists do
            layer.contents ← layer.contents + page
            page ← getNextPage()
        end
        layer.contents ← layer.contents + page
        layerTable[layer.id] ← layer
    end
end
```

Since this methodology can modify G-code at the instruction level, it is possible to execute fine-grained attacks that significantly degrade printed parts in a manner that is undetectible by visual inspection and other nondestructive testing methods.

5. G-Code Toolpath File Attacks

This section discusses the temperature modification and infill exclusion attacks that were performed against G-code generated for fused filament fabrication printers. The attacks were chosen for their ability to sabotage printed parts as well as print environments [17], demonstrating the serious impacts of G-code manipulation.

5.1 Temperature Modification Attacks

A temperature modification attack modifies the extrusion head temperature during printing. The temperature at which a part is printed determines its physical properties such as layer adhesion and material phase transformation. An extreme temperature modification from the baseline temperature for a given material can cause air gaps to form between printed layers that are visible to the naked eye.

A temperature modification attack also impacts the print environment. Altering the extruder head temperature affects the printer itself.

Prolonged printer operation outside its normal parameters can induce printer component wear, material blockage and premature breakage.

5.2 Infill Exclusion Attacks

An infill exclusion attack alters a printed part by selectively removing material from specified layers. Printed parts may not be completely solid; often they are printed as shells to conserve printing material (filament). In such cases, infill with a pattern such as stars or squares fills the 3D-printed shell. The infill pattern and density affect part strength and durability, so modifying the infill via G-code manipulation impacts the mechanical properties of the part.

It can be difficult to detect infill changes by visually inspecting a completed part. Additionally, removing small percentages of infill from part layers can reduce part strength without a significant reduction in weight, making the modification difficult to detect via nondestructive testing. Indeed, by targeting specific layers of a part during printing, critical areas of the final part can be compromised.

6. Attack Results and Mitigations

This section describes the results of experiments that used the toolkit to launch slicing software execution attacks on two software suites that create G-code toolpath files for a variety of fused filament fabrication printers. The toolkit, written in C, runs on the same Ubuntu 20.04 virtual machine as the slicing software suites. Since the software execution attack has root access, the toolkit code is able to search for the slicer process ID and attach to the runtime process image in order to scan and modify the memory.

For each slicing software suite, an attack was executed against the first slice on a clean boot of the virtual machine. After the execution, the modified G-code toolpath file was saved for transfer to a printer using an SD card. Modified G-code toolpath files in the temperature modification experiments were submitted to an Ender 3 fused filament fabrication printer. Modified G-code toolpath files in the infill exclusion experiments were submitted to a Prusa i3 Mk3S fused filament fabrication printer. Both the printers employed polylactic acid (PLA) filament as the printing material.

6.1 Attack Effectiveness Experiments

The first set of experiments evaluated toolkit performance. The execution time metric, corresponding to the duration of a successful attack, was used to evaluate attack detectibility based on its impact on user

Table 1. Average attack execution times over ten runs.

Attack	Software	Execution Time
Temperature Modification	Slicer Alpha	22 ms
Temperature Modification	Slicer Beta	32 ms
Infill Exclusion	Slicer Beta	184 ms

experience. An attack attaches to the runtime process image, which freezes slicer software execution, impacting user experience and potentially raising an alarm. In the experiments, temperature modification attacks were executed on G-code generated by slicers Alpha and Beta whereas infill exclusion attacks were only executed on G-code generated by slicer Alpha. The reason for not executing infill exclusion attacks on slicer Alpha is explained below.

Table 1 shows the average durations of the two attacks. The average attack execution times obtained for the temperature modification attacks on Alpha and Beta G-code were 22 ms and 32 ms, respectively. The average attack execution time obtained for the infill exclusion attack on slicer Beta G-code was 184 ms. The low execution times are expected for the temperature modification attacks because only two lines of G-code in two different layers need to be modified to successfully implement the attacks. In contrast, the infill exclusion attacks require considerable time because every line of G-code in multiple layers is modified. Nevertheless, the 22 ms to 184 ms attack execution window is well within the acceptable response time [12], so attack execution would not change the normal user experience. The attacks were also surreptitious because the firmware in the two printers did not raise any exceptions or warnings about malformed G-code produced by the two slicing software suites.

The second set of toolkit performance experiments evaluated the ability of the toolkit to identify G-code in memory corresponding to individual print layers. This is important because the greater the percentage of individual layers detected at runtime, the finer the granularity and more insidious the attacks.

Table 2 shows the average percentages of G-code layers identified for the two slicing software suites. Exceptional G-code layer identification of 99.73% was obtained for slicer Beta. The G-code layer identification of 41.29% for slicer Alpha is modest, but the temperature attacks were, nevertheless, successful. Note that the low layer identification percentage obtained for the G-code generated by slicer Alpha renders infill exclusion

Table 2. Average G-code layers identified over ten runs.

Software	**G-Code Layers**
Slicer Alpha	41.29%
Slicer Beta	99.73%

attacks infeasible. This is because the attacks require every line of G-code in multiple layers to be modified.

6.2 Temperature Modification Experiments

Temperature modification attacks were launched against G-code toolpath files generated by slicer Alpha. In the attacks, the extruder head temperature was reduced by 8°C from the normal operating temperature of 198°C while printing just seven centrally-located layers of the 530 total layers of the parts before being returned to the original temperature.

The printer did not indicate any issues with the lower temperature during printing. No discernible pauses in printing occurred while the temperature was decreased and increased. The temperature modifications did not result in any observable differences in the printed parts.

Tensile tests were performed on parts that were printed in the standard ASTM dogbone shape. However, the test samples were modified slightly by printing two holes near the two ends to mount them on a servohydraulic tensile testing system. Ten control samples were printed at the temperature of 198°C and ten attack samples were printed with the 8°C drop in temperature for seven centrally-located layers.

Table 3. Tensile test results for the temperature modification attacks.

Sample (Temperature)	**Average Breaking Force**	**Standard Deviation**	**Strength Reduction**	**P(T≤t) Two-Tailed Test**
Control (198°C)	964.9 N	72.1 N	–	–
Attack (190°C)	829.7 N	29.2 N	14.0%	0.00019

Table 3 shows the results of the tensile tests on the printed samples. The average breaking forces for the control and attack samples were 964.9 N and 829.7 N, respectively. This corresponds to a 14% reduction in the average tensile strength of parts due to the temperature modification attacks. A two-tailed t-test indicated a statistically-significant

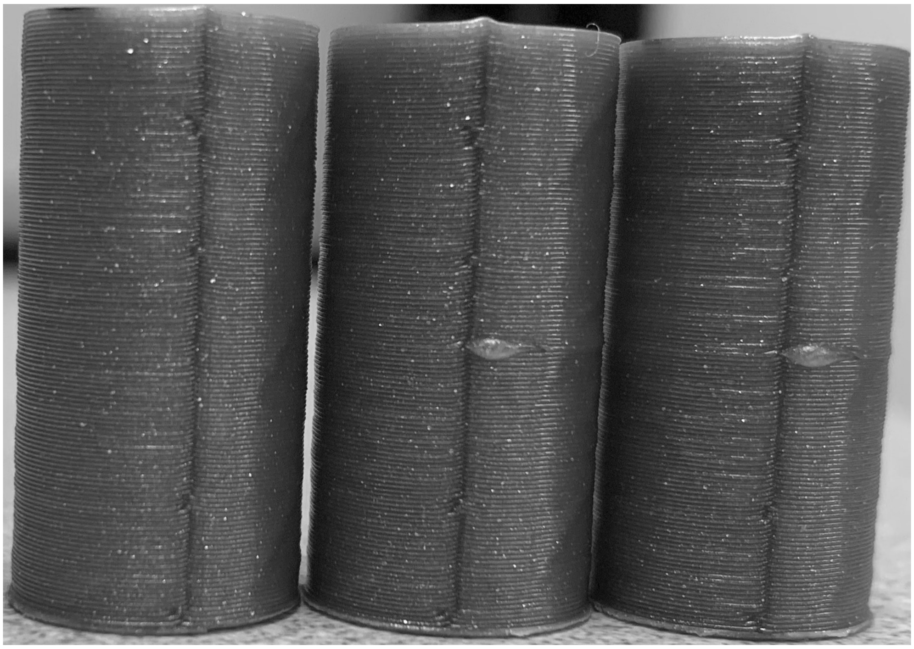

Figure 3. Control and attack samples in the infill exclusion experiments.

difference between the attack and control sample populations. Examination of the attack samples revealed that all the samples failed at the seven layers that were printed when the temperature was reduced.

6.3 Infill Exclusion Experiments

The infill exclusion experiments employed solid ASTM cylinders with 6.35 mm radius and 25.4 mm height printed using G-code toolpath files generated by slicer Beta. Figure 3 shows three sample prints. The cylinder on the left is a control sample. The attack sample in the center had infill excluded from five centrally-located layers of the 127 total layers. The attack sample on the right had infill excluded from 25 centrally-located layers of the 127 total layers.

Printing the control and attack samples took the same amount of time because the extruder head went through the same motions for all the samples, except that no infill was printed in some layers of the attack samples. The reduction in mass due to infill exclusion was negligible. As seen in Figure 3, the only discernible differences are small blobs of extra filament at the attacked layers. However, it is common for parts printed

Table 4. Compresssion test results for the infill exclusion attacks.

Sample (Layers Removed)	**Average Failure Force**	**Standard Deviation**	**Strength Reduction**	**P(T≤t) Two-Tailed Test**	**Average Mass**
Control (0)	2,446.7 N	130.9 N	–	–	2 g
Attack (5)	2,187.2 N	53.8 N	10.6%	0.0082	2 g
Attack (25)	1,959.5 N	233.9 N	19.9%	0.00018	1.9 g

via fused filament fabrication to have small masses of extra filament that are broken off or sanded down during the post-processing step.

The attacked cylinders were evaluated by performing compression tests using a universal testing machine. In the experiments, five control samples and 20 attack samples (ten for each of the two attacks) were compressed with increasing force until failure.

Table 4 shows the compression test results. Excluding infill from five of the 127 layers resulted in a 10.6% decrease in the average compressive strength of the printed parts. As expected, excluding infill from 25 of the 127 layers resulted in a significant decrease of 19.9% in the average compressive strength. Two-tailed t-tests indicated statistically-significant differences between each attack sample population and the control sample population.

Figure 4 shows the average compression failure curves for the control and attack samples. Note that the peaks in the curves correspond to the points of part failure. As expected, the average failure force for the control samples is higher than the average failure force for the attack samples. Moreover, the greater the number of layers with excluded infill, the lower the average failure force.

6.4 Discussion

The experimental results demonstrate that slicing software attacks weaken printed parts with little or no discernible differences. Furthermore, temperature modification and infill exclusion are just two of many attacks on G-code in toolpath files. G-code specifies print speed, fan speed and other printer parameters, all of which affect the properties of printed parts. Manipulating printer parameters by modifying G-code could also damage the printer itself.

Executing the slicing software attacks developed in this research requires root access, but it is an attractive attack option despite the many opportunities offered by arbitrary code execution. This is because the

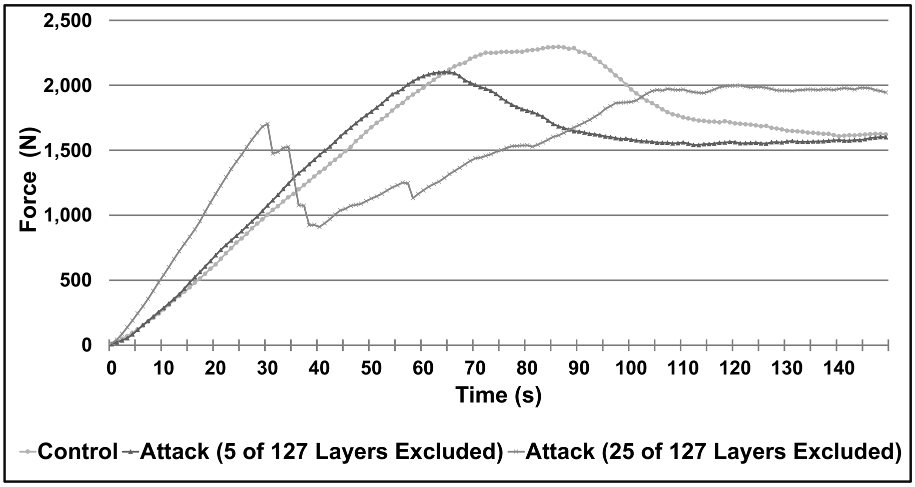

Figure 4. Average failure curves for the infill exclusion attacks.

software execution exploit can be executed and operated autonomously, without the need for Internet access. Also, the current toolkit could be reconfigured as malware that runs in the background and constantly scans runtime process images and manipulates G-code. Although targeted attacks on parts would produce more extreme effects on printed products, the malware would not need to know the precise parts being printed in order to launch attacks such as temperature modification and infill exclusion. Indeed, the malware could run independently and attack parts and print environments with little or no human intervention. Thus, slicing software exploitation can result in surreptitious attacks that are difficult to detect and mitigate.

The main limitations of the slicing software execution attacks are that they apply to ASCII-encoded G-code and require G-code to be extracted from heap memory. As a result, the attacks only target slicers that store dynamic copies of toolpath files. However, this characteristic is common in slicers that permit users to send G-code toolpath files directly to printers and to dynamically manipulate G-code toolpath settings.

Another limitation is that certain layers of G-code were not recoverable in the case of slicer Alpha. Future research will employ reverse engineering to investigate this anomaly and create fine-grained attacks.

Finally, a limitation with direct memory modification is one-to-one byte replacement. This increases the creativity needed to craft attacks. Of course, attacks against G-code could be performed with more freedom using other attack vectors such as direct access to a G-code toolpath file via the network or controller computer. These opportunities eliminate the need to exploit a memory vulnerability, rendering G-code attacks widely applicable and a major concern.

7. Mitigations

In the experiments, the attacks that modified ASCII-encoded G-code in heap memory leveraged access to slicing software from a controller computer. However, the same exploits could be applied by leveraging access to slicing software from a network or access to a slicing software update. The attack impacts include intellectual property theft as well as part and print environment sabotage.

The first set of mitigations should combat the attack vectors that provide access to a controller computer, network and slicing software update. These are accomplished by instituting strong user authentication and access controls on the controller computer and network, and requiring signed and encrypted slicing software updates. If an attacker breaches these defenses, the next set of defenses should protect ASCII-encoded G-code in heap memory. This is accomplished by obfuscating the G-code in heap memory, which would make it difficult to identify the toolpath layers. Another mitigation technique is to detect and block common behavior sequences (such as scanning runtime process image memory) that occur when attempts are made to identify, extract and modify G-code.

However, G-code attacks beyond modifying code images in heap memory could be launched by leveraging other attack vectors and other vulnerabilities. This emphasizes the need to conduct a comprehensive analysis of the attack vectors, targets, target vulnerabilities and attacks that enable G-code manipulation, along with countermeasures for combating the attack vectors that provide access to targets and attacks that exploit the identified vulnerabilities.

8. Conclusions

3D printing is commonly used to create mission-critical parts in the energy, healthcare, transportation and defense sectors. A 2021 cyber security audit of U.S. Department of Defense additive manufacturing sites determined that all the reviewed sites did not consistently manage or secure their systems to prevent unauthorized changes and ensure design data integrity. Since additive manufacturing is constantly exposed to cyber threats, it is imperative to continually analyze the attack surface, identify vulnerabilities, devise exploits and institute countermeasures.

This research has identified a novel slicing software vulnerability where G-code corresponding to part slices is stored as plaintext ASCII characters in heap memory during execution. The vulnerability was discovered in two open-source, full-featured slicing software suites that support many 3D printers. Exploiting the slicing software vulnerability requires full code execution privileges (root access) to the controller machine that executes the slicing software. The attacks access the runtime process image, scan the image memory, extract the plaintext G-code, perform various modifications to the G-code and write the code back to image memory.

The temperature modification and infill exclusion attacks demonstrate the ability to sabotage printed parts and print environments. The temperature modification attacks, which reduced the extruder head temperature by just 8°C while printing less than 1.5% of the part layers, resulted in a 14% drop in the average tensile strength. The infill exclusion attacks, which omitted infill from less than 4% of the part layers, reduced the average compressive strength by 10.6%.

Although the discovered vulnerability can be mitigated using strong authentication and access controls along with G-code obfuscation, the ability to automate surreptitious, fine-grained attacks that significantly degrade printed parts in ways that are imperceptible to the human eye and undetectible by nondestructive testing methods is a serious concern. Clearly, strong, systematic efforts must be directed at securing additive manufacturing systems from cyber threats.

Acknowledgement

This research was supported by the National Science Foundation under Grant no. DGE 1501177 and by UT-Battelle under Contract no. DE-AC05-00OR22725 with the U.S. Department of Energy.

References

[1] 3MF Consortium, 3MF Specification, San Francisco, California (`3mf.io/specification`), 2020.

[2] S. Belikovetsky, M. Yampolskiy, J. Toh, J. Gatlin and Y. Elovici, dr0wned – Cyber-physical attack with additive manufacturing, presented at the *Eleventh USENIX Workshop on Offensive Technologies*, 2017.

[3] I. Birrell, 3D-printed prosthetic limbs: The next revolution in medicine, *The Guardian*, February 19, 2017.

[4] J. Burke, 3D printing off to the races, *Oak Ridge National Laboratory Blog*, Oak Ridge National Laboratory, Oak Ridge, Tennessee (`www.ornl.gov/blog/3d-printing-races`), April 26, 2019.

[5] J. Ellis, 3D-printed nuclear reactor promises faster, more economical path to nuclear energy, *Oak Ridge National Laboratory News*, Oak Ridge National Laboratory, Oak Ridge, Tennessee (`www.ornl.gov/news/3d-printed-nuclear-reactor-promises-faster-more-economical-path-nuclear-energy`), May 11, 2020.

[6] S. Ford, Additive manufacturing technology: Potential implications for U.S. manufacturing competitiveness, *Journal of International Commerce and Economics*, vol. 6(1), pp. 40–74, 2014.

[7] Inspector General, U.S. Department of Defense, Audit of the Cybersecurity of Department of Defense Additive Manufacturing Systems, Washington, DC (`media.defense.gov/2021/Jul/07/2002757308/-1/-1/1/DODIG-2021-098.PDF`), 2021.

[8] J. Keller, The navy can now 3D-print submarines on the fly for SEALs, *Task and Purpose* (`taskandpurpose.com/gear-tech/navy-3d-printing-submarines`), July 31, 2017.

[9] T. Kramer, F. Proctor and E. Messina, The NIST RS274NGC Interpreter – Version 3, NIST Interagency/Internal Report 6556, National Institute of Standards and Technology, Gaithersburg, Maryland, 2000.

[10] Library of Congress, STL (Stereolithography) File Format Family, Washington, DC (`www.loc.gov/preservation/digital/formats/fdd/fdd000504.shtml`), September 9, 2019.

[11] T. McCue, Additive manufacturing industry grows to almost $12 billion in 2019, *Forbes*, May 8, 2020.

[12] R. Miller, Response time in man-computer conversational transactions, *Proceedings of the AFIPS Fall Joint Computer Conference, Part I*, pp. 267–277, 1968.

[13] S. Moore, P. Armstrong, T. McDonald and M. Yampolskiy, Vulnerability analysis of desktop 3D printer software, *Proceedings of the 2016 Resilience Week*, pp. 46–51, 2016.

[14] S. Moore, W. Glisson and M. Yampolskiy, Implications of malicious 3D printer firmware, *Proceedings of the Fiftieth Hawaii International Conference on System Sciences*, 2017.

[15] H. Pearce, K. Yanamandra, N. Gupta and R. Karri, FLAW3D: A Trojan-Based Cyber Attack on the Physical Outcomes of Additive Manufacturing, arXiv: 2104.09562 (`arxiv.org/abs/2104.09562`), 2021.

[16] B. Post, B. Richardson, P. Lloyd, L. Love, S. Nolet and J. Hannan, Additive Manufacturing of Wind Turbine Molds, Document ORNL/TM-2017/290, Oak Ridge National Laboratory, Oak Ridge, Tennessee, 2017.

[17] M. Rais, Y. Li and I. Ahmed, Dynamic thermal and localized filament kinetic attacks on a fused-filament-fabrication-based 3D printing process, *Additive Manufacturing*, vol. 46, article no. 102200, 2021.

[18] L. Sturm, C. Williams, J. Camelio, J. White and R. Parker, Cyber-physical vulnerabilities in additive manufacturing systems: A case study attack on the .STL file with human subjects, *Journal of Manufacturing Systems*, vol. 44(1), pp. 154–164, 2017.

[19] C. Xiao, Security attack on 3D printing, presented at the *xFocus Security Conference* (`www.claudxiao.net/Attack3DPrinting-Claud-en.pdf`), 2013.

[20] M. Yampolskiy, W. King, J. Gatlin, S. Belikovetsky, A. Brown, A. Skejellum and Y. Elovici, Security of additive manufacturing: Attack taxonomy and survey, *Additive Manufacturing*, vol. 21, pp. 431–457, 2018.

[21] S. Zeltmann, N. Gupta, N. Tsoutsos, M. Maniatakos, J. Rajendran and R. Karri, Manufacturing and security challenges in 3D printing, *Journal of the Minerals, Metals and Materials Society*, vol. 68(7), pp. 1872–1881, 2016.