4 Reliable Code Execution on a Tamagotchi

by Natalie Silvanovich

Tamagotchis are an excellent target for reverse engineering for a number of reasons: They have a limited number of inputs and outputs, they run on a poorly documented 6502 microcontroller and they're, well, Tamagotchis. Recently, I discovered a technique for reliably executing foreign code on a Tamagotchi.

Let's begin at the beginning. Modern Tamagotchis run on a GeneralPlus GPLB52X LCD controller, a lightweight 6502 controller that uses an internal mask ROM for all code and some data. This means that exploitation is necessary to free the Tamagotchi from the shackles of its read-only code. Also, in the absence of any debug outputs, code execution provides valuable insight into the internals of the Tamagotchi and its MCU.

There are four inputs into a Tamagotchi that can be manipulated by the user. (1) The buttons, (2) the EEPROM that saves the Tamagotchi state across resets, (3) the IR interface and (4) certain accessories containing external SPI memory called figures. Attempts to find useful bugs in the EEPROM and IR interface were unsuccessful, so I moved onto the figures. Eventually I found an exploitable bug in how the Tamagotchi processes figure data.

When attached to a Tamagotchi, figures add extra functionality, such as games or items. So attaching a figure might allow your Tamagotchi to play shuffleboard, purchase a vacuum cleaner or attend 30c3. The bug I found was in the processing of game data. Game logic is not actually included in the figure data; rather, the figure provides an index to the game logic in the Tamagotchi's mask ROM.[5] Changing this index causes some very strange behavior. If the index is an expected value, from 0 to about 0x20, a game will be played as expected, but for higher indexes, the device will freeze, requiring a reset. Even stranger, if the index is very high (0xD8 or higher), the Tamagotchi jumps to a different, valid screen, such as feeding the Tamagotchi or giving it a bath, and the Tamagotchi functions normally afterwards. This made me suspect that the game index was used as an index into a jump table and that freezing was due to jumping to an invalid location.

With no way to gain additional information about the cause of the behavior, and about 200 possible vulnerabilities, it made sense to to fill up as much memory as possible up with a NOP sled, try all possible indexes, and hope that one caused a jump to the right location. Unfortunately, the only memory controllable by the figure is the LCD RAM, so I filled that with NOPs and shellcode. (The screen data starts at 0x1C80 in the figure memory, and maps to 0x1000 in the Tamagotchi memory, for people trying this at home.) After several tries and some fiddling the shellcode, index 0xD4 lead to very unreliable code execution. This code execution allowed me to perform a complete ROM dump of the Tamagotchi, which in turn led to the ability to better analyze the bug.

The following code contains the vulnerability. Please note that the current state (current_state_22) is set from the game index without validation.

```
seg004:4E2E    LDA    byte_1A4
seg004:4E31    BEQ    loc_44E39
seg004:4E33    LDA    gameindex2
seg004:4E36    JMP    loc_44E3C
seg004:4E39    LDA    gameindex1
seg004:4E3C    CLC
seg004:4E3D    ADC    #$27 ;
seg004:4E3F    STA    current_state_22
seg004:4E41    JMP    locret_44E4C
```

[5]The important index is located at address 0x18 in figure memory.

The main Tamagotchi execution loop checks the state based on a timer interrupt, then makes a state transition if the state has changed. The state transition is as follows.

```
ROM:EFE8    LDX     current_state_22
ROM:EFEA    LDA     $F00E,X
ROM:EFED    STA     change_page
ROM:EFF0    STA     current_page
ROM:EFF2    BEQ     loc_F001
ROM:EFF4    LDA     #0
ROM:EFF6    STA     off_34
ROM:EFF8    LDA     #$40  ;  '@'
ROM:EFFA    STA     off_34+1
ROM:EFFC    LDA     current_state_22
ROM:EFFE    JMP     (off_34)
```

In essence, the Tamagotchi looks up the page of the state in a table at 0xF00E, then jumps to address 0x4000 in that page. Looking at this code, it is clear why my first exploit was unreliable. 0xD4 + 0xF00E + 0x27 is 0xF109, which resolves to a value of 0x3c. Since the Tamagotchi only has 19 pages, this is an invalid page number. Testing what would happen if the MCU was provided an invalid page, addresses 0x4000 and up resolved to 0xFF.

This means that there are two possibilities of how this exploit works. Either the memory addresses are floating and sometimes end up with values that, when executed, send the instruction pointer to the LCD RAM, or the undefined instruction 0xFF, when executed, puts the instruction pointer into the right place, sometimes. Barring bizarreness beyond my wildest imagination, neither of these possibilities would allow for the exploit to be made more reliable though manipulation of the figure data.

Instead, I looked for a better index to use, which turned out to be 0xCD. 0xCD + 0xF00E + 0x27 is 0xF102, which maps to part of the LCD segment table, which has a value of 4. Jumping to 0x4000 in page 4 immediately indexes into another page table.

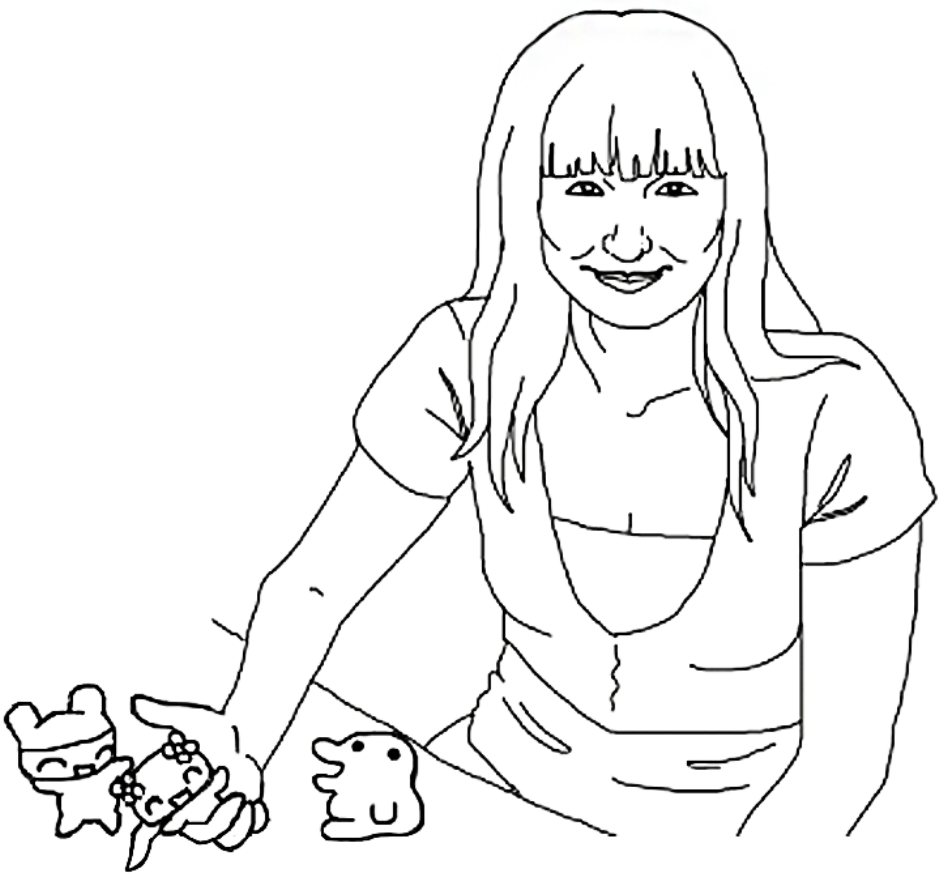

```
seg004:4000     LDA     #$D
seg004:4002     STA     $34
seg004:4004     LDA     #$40  ;  '@'
seg004:4006     STA     $35
seg004:4008     LDA     $22
seg004:400A     JMP     jump_into_table_D27F
```

This index is also out of range, and indexes into a code section:

```
seg004:41F5     INC     $11E
```

Interpreted as a pointer, however, this value is 0x1EEE. The LCD RAM range is from 0x1000 to 0x1200, but fortunately, bits 2-7 of the upper byte of addresses in the 0x1000-0x2000 range are ignored, so reading 0x1EEE returns the value at 0x10EE. This means that playing a game with the index of 0xCD will execute code in the LCD RAM every time!

While reading POC||GTFO obligates you to share a copy with a neighbour, trying this on your own Tamagotchi is only strongly recommended. Further instructions can be found by unzipping the PDF of this issue.



"The ancient teachers of this science promised impossibilities and performed nothing. The modern masters promise very little; they know that metals cannot be transmuted and that the elixir of life is a chimera but these philosophers, whose hands seem only made to dabble in dirt, and their eyes to pore over the microscope or crucible, have indeed performed miracles. They penetrate into the recesses of nature and show how she works in her hiding-places. They ascend into the heavens; they have discovered how the blood circulates, and the nature of the air we breathe. They have acquired new and almost unlimited powers; they can command the thunders of heaven, mimic the earthquake, and even mock the invisible world with its own shadows." – Shelley 3:16