

MTS Finds Ridiculous RG Exploit

Adam Fraser-Kruck



I love building stuff!

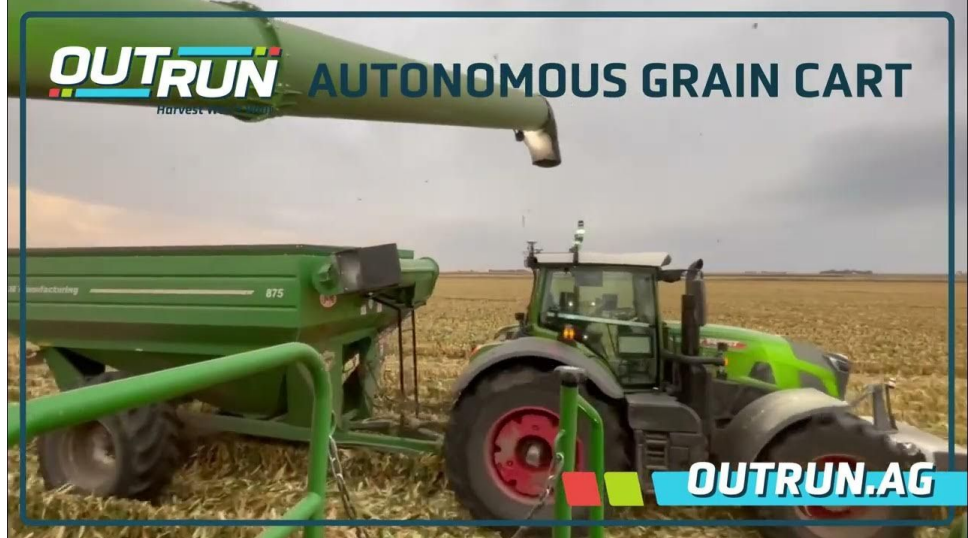
Large ramp in backyard.

Skateboard basement press.

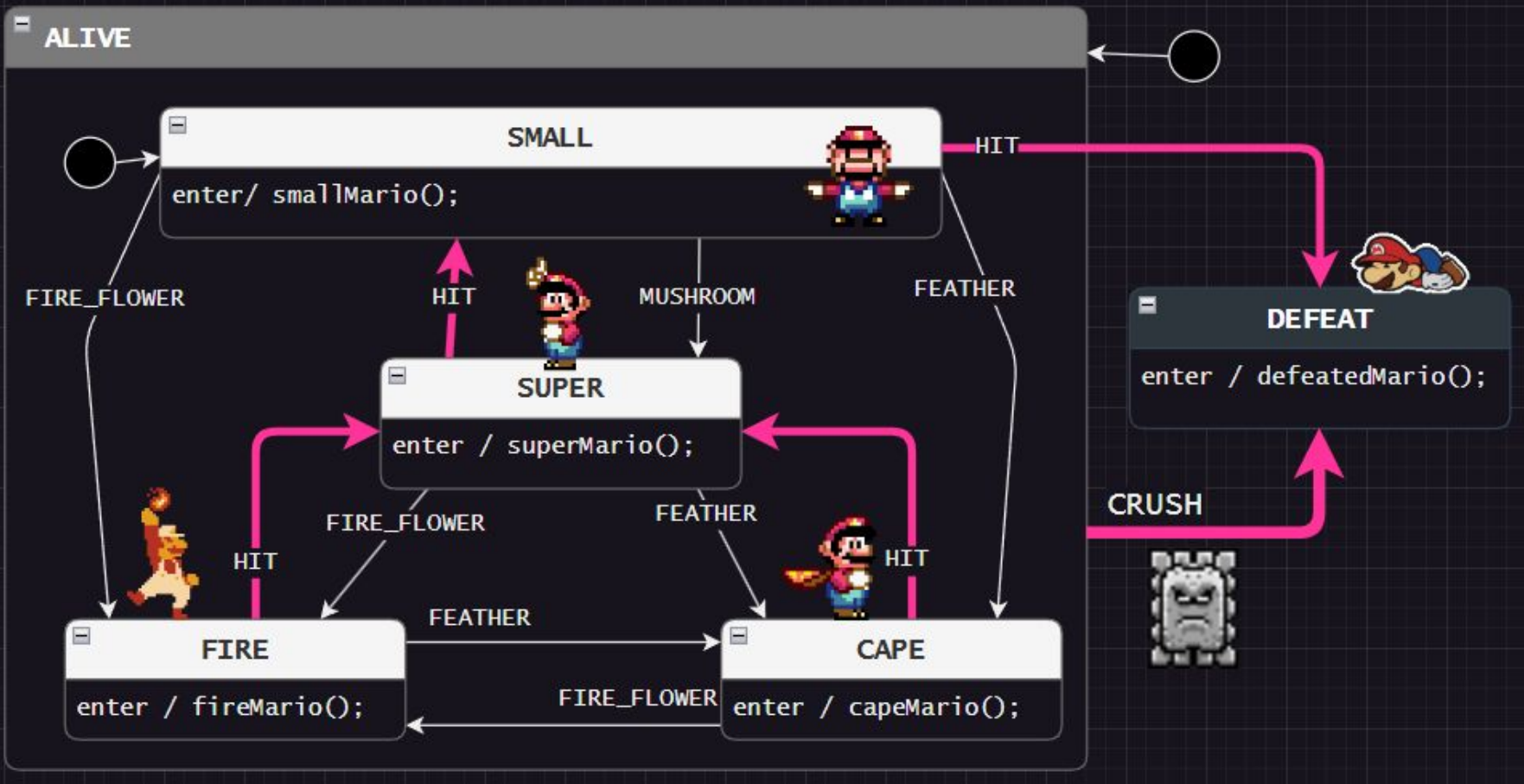
Circa 2001

I love building stuff!
Embedded consulting 2002
Wireless Basemaster System





Senior Embedded Developer
PTx Trimble (8 years)



I love building stuff! State Machines ❤️

MTS - Technology Development

15 years ago I worked for an ISP mostly creating automated tests for equipment.

Before MTS would buy thousands of a product, we'd make sure it was good.

Changing names: I want to share interesting bugs/exploits, not drag a company.

Low Cost RG → Ridiculous Security

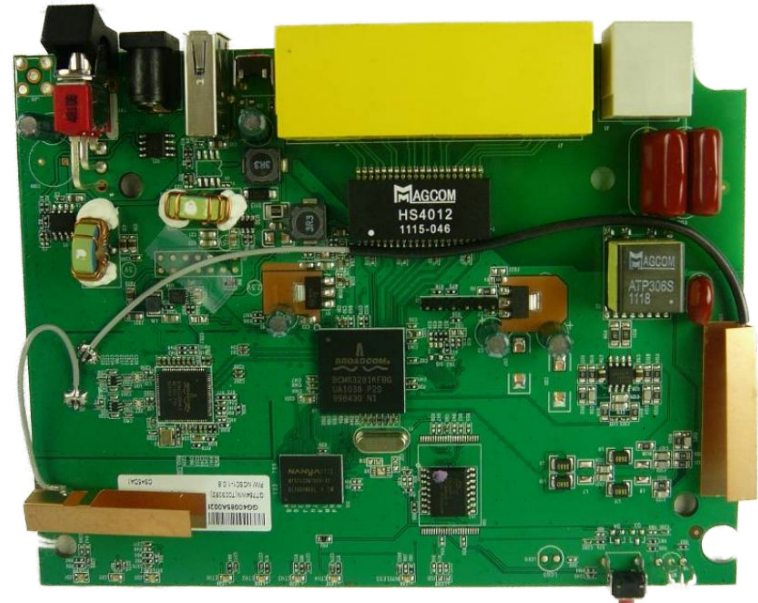
One residential gateway (RG) from a manufacturer (let's call SuperTech) was really fun to test. SuperTech sells worldwide in the millions. Big company.

Why was it fun?

It was ripe with issues! Felt good to find them :)

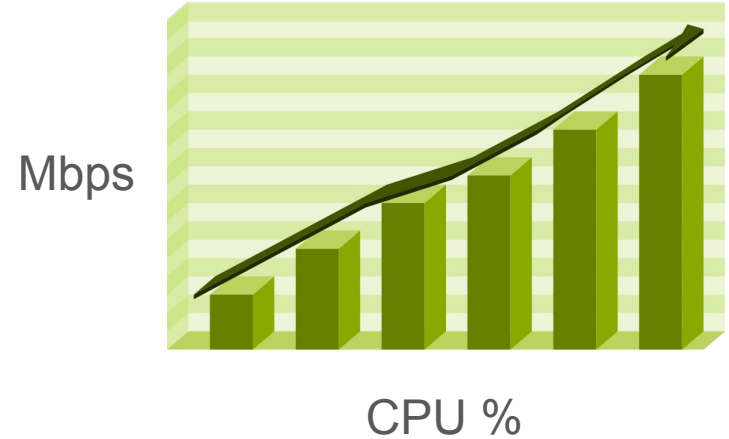
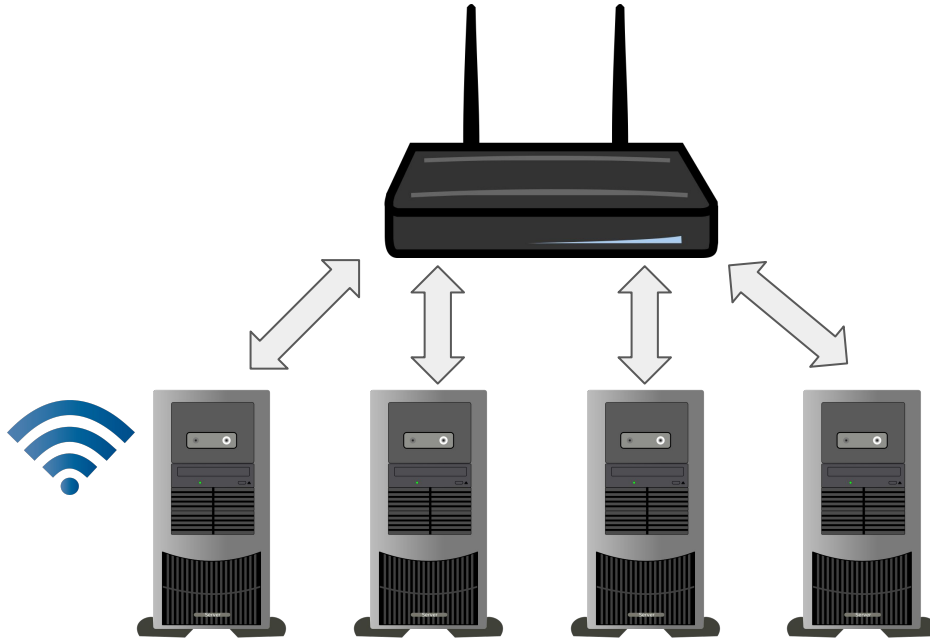
This was very surprising because it was already being distributed by Verizon.

Surely little MTS wasn't finding show stoppers!?



One Test Setup

Most of my testing was black box (found mostly minor stuff), but we were also given root access.



Browsing The File System

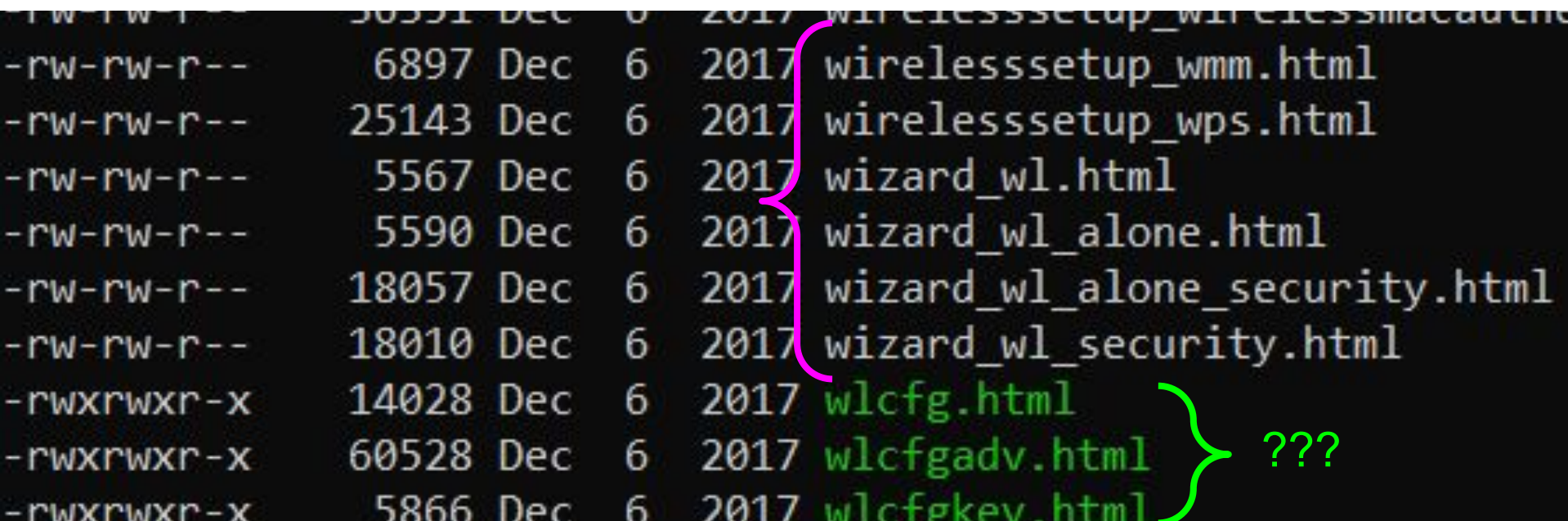
```
# ls /webs/
```

http://192.168.0.1/wizard_wl.html

<http://192.168.0.1/wlcfg.html>

Guess what I got?

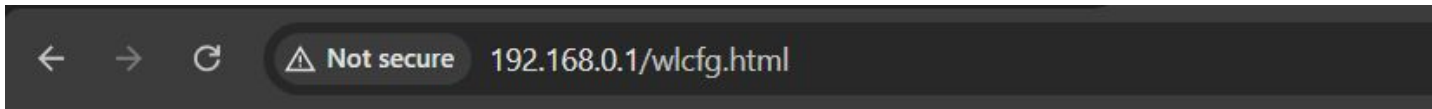
```
drwxrwxr-x  2 30551 Dec  6 2017 wirelesssetup_wirelessmacauth
-rw-rw-r--  1 6897 Dec  6 2017 wirelesssetup_wmm.html
-rw-rw-r--  1 25143 Dec  6 2017 wirelesssetup_wps.html
-rw-rw-r--  1 5567 Dec  6 2017 wizard_wl.html
-rw-rw-r--  1 5590 Dec  6 2017 wizard_wl_alone.html
-rw-rw-r--  1 18057 Dec  6 2017 wizard_wl_alone_security.html
-rw-rw-r--  1 18010 Dec  6 2017 wizard_wl_security.html
-rwxrwxr-x  1 14028 Dec  6 2017 wlcfg.html
-rwxrwxr-x  1 60528 Dec  6 2017 wlcfgadv.html
-rwxrwxr-x  1 5866 Dec  6 2017 wlcfgkey.html
```



Unsecured Broadcom Sample Page!

160+ pages!

Ripe for attack



Wireless -- Basic

This page allows you to configure basic features of the wireless LAN interface. You can enable or disable the on country requirements.

Click "Apply/Save" to configure the basic wireless options.

- Enable Wireless
- Hide Access Point
- Clients Isolation
- Disable WMM Advertise
- Enable Wireless Multicast Forwarding (WMF)

SSID:

BSSID: 00:26:B8:C6:43:71

Country:

Max

Obligatory Meme



Let's send an email

Not everyone at MTS initially thought this was a big deal.

"Bah! An attacker needs physical or secured Wi-Fi access."

So I sent an email to another person testing this RG. Something like:

Hi Gav,

I've just changed your Wi-Fi SSID.

```
<img src="http://192.168.0.1/wlcfg.html?ssid=8HzWanIp&..."
```

Have a great day!

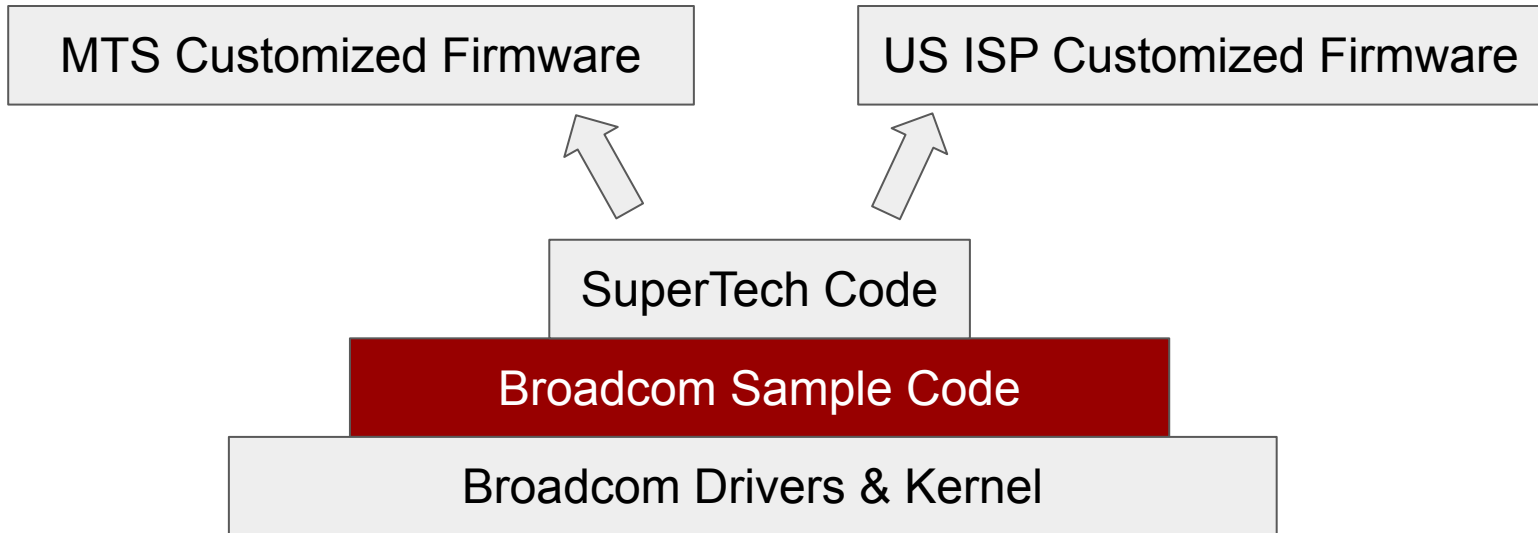
Super susceptible to Cross Site Request Forgery (CSRF) attacks.

Engineering Disillusion

I couldn't believe it. Where was the engineering responsibility?

Sample code is often **incomplete**, doesn't consider security or error handling...

It (ideally) should **not** form the **core** of your production firmware.



The Threat Is Real

Changing Wi-Fi settings could cause some problems, but there were many other vulnerable settings.

DNS for example.

Bad DNS Entries + Banking = Stolen Money

A few years earlier 2wire RGs & an ISP seriously affected 2 million users.

An email CSRF attack would poison DNS entries for Mexican bank sites.

The Mexican ISP customization had no password by default!!!



User receives an email with exploit codes and a link to a malicious Web site

Exploit code takes advantage of a vulnerability in 2Wire Modems, which allows an attacker to modify the local DNS servers and hosts

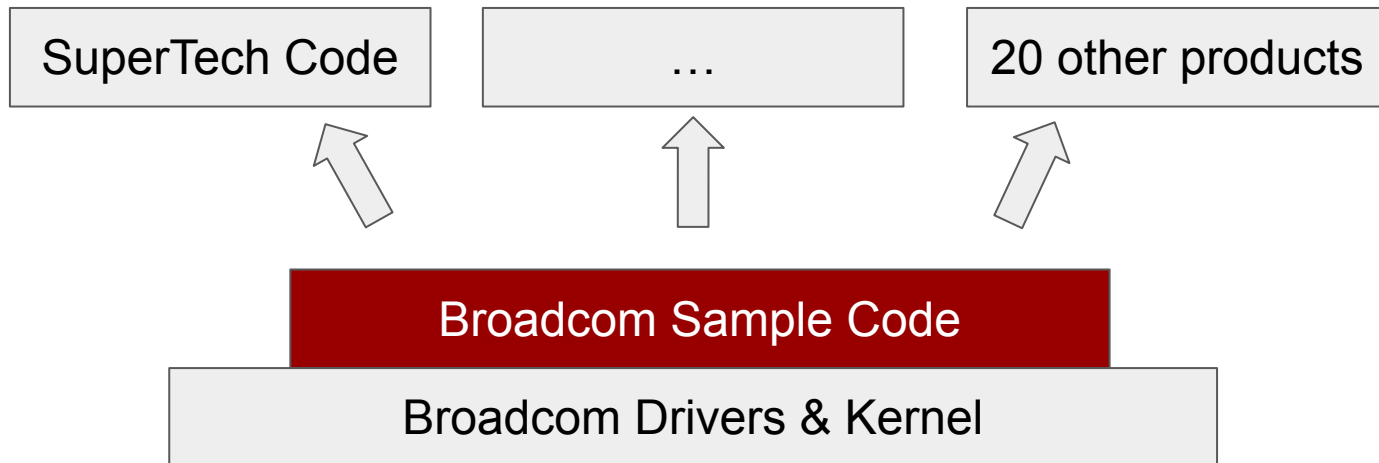
Side Note

While the SuperTech product did have a lot of issues, they were very quick fixing the issues and were pleasant to work with.

All of the exploits shown here were patched long ago, but similar mistakes may happen again.

How To Apply Today

1. Scan for hidden sample management pages
2. Find exploits in other devices using same chipset
 - a. Use techinfodepot.shoutwiki.com or deviwiki.com to find CPU Chipset
 - b. Check for exploits in other devices (may share same code)
 - c. Might be easier to get source code or root other devices.



Bonus slides - State Machines & Security

Anyone familiar with state machines?

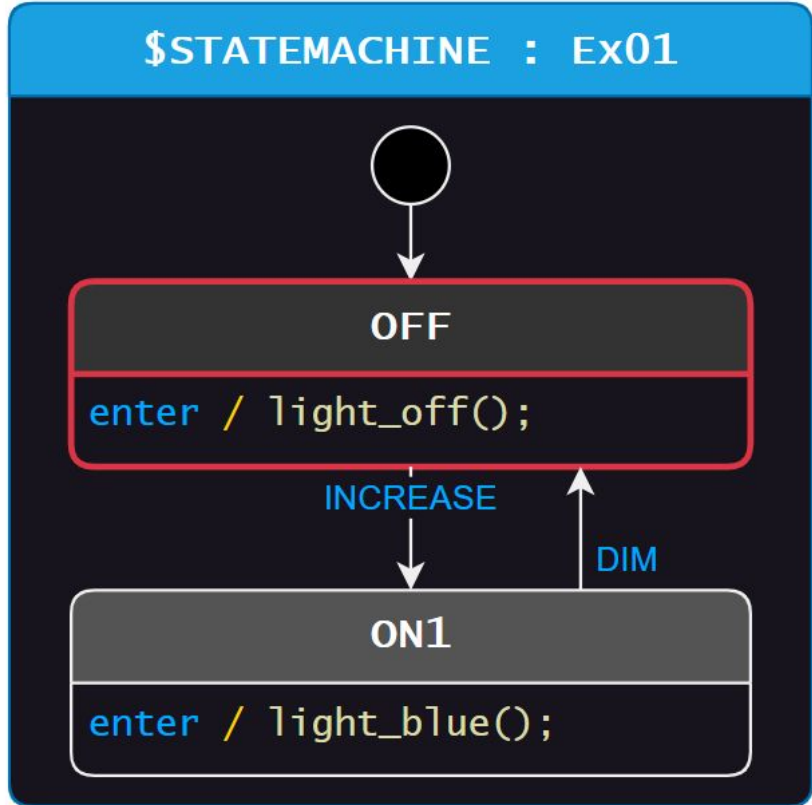
State Machines?

What is a state machine?

- Hardware or software
- Has states
- Has transitions between states
- Has actions

Other names:

- Finite State Machine (**FSM**)
- State Chart
- State Pattern



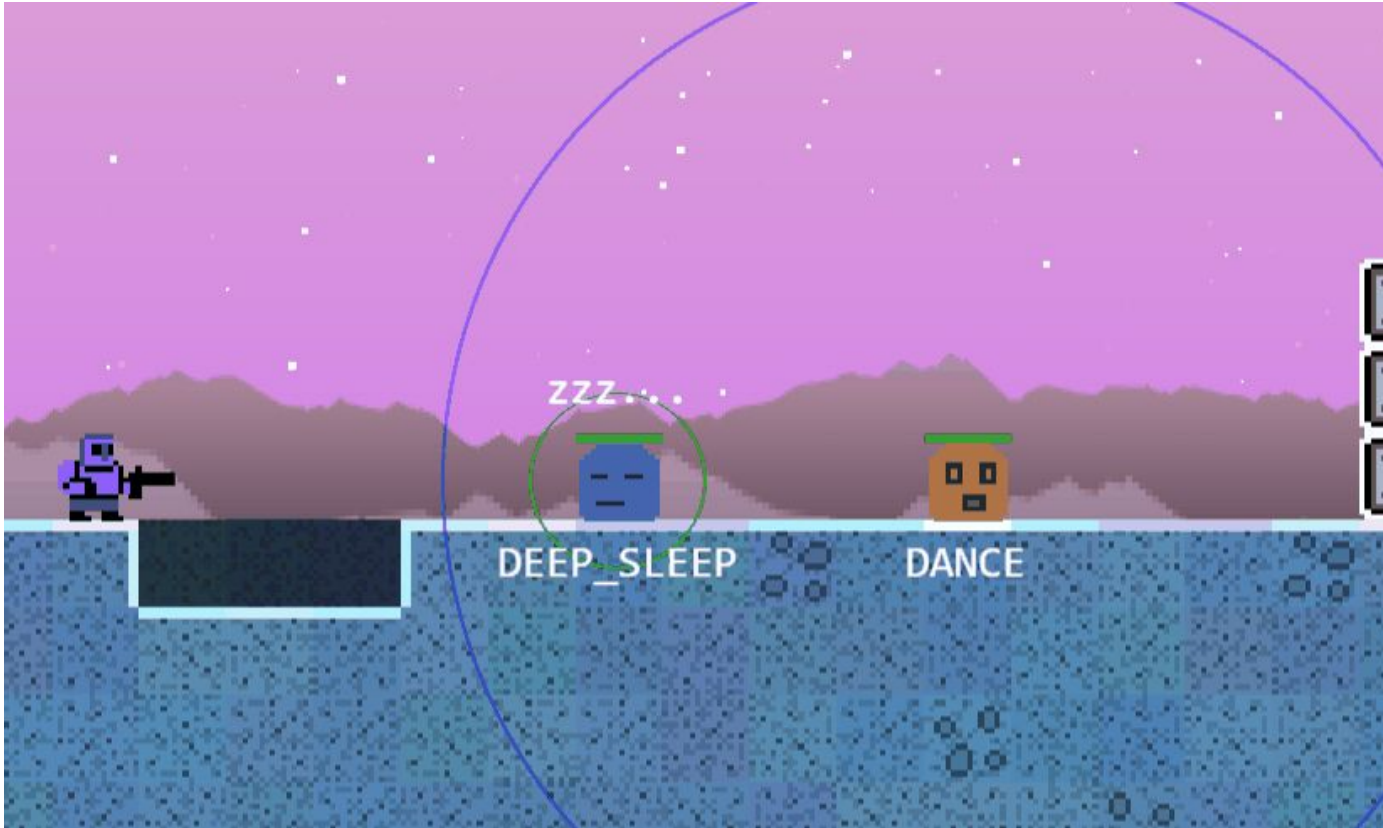
Visual State Machines With StateSmith

I'm working on an open source project **StateSmith** which helps solve many problems. Not feature complete yet, but already very useful.



Many IoT devices, a well known vacuum brand, safety critical application in France, self driving vehicles in South Korea.

🌐 Protocols, 📱 User Interfaces, 💻 Drivers/Embedded, 🚀 Control Systems,
🎮 Video Games, 🤖 Robotics, 💠 Workflow Process, 📺 Parsers/Decoders, ...



No FSM Trap - Evolve Code & Trust Events (Firmware)

We start with a simple application that receives message events.

Then a new requirement is added. Developer follows existing pattern. **TRUST!**

```
on_event_1() { x = 10 }  
on_event_2() { show "ABC" }  
...  
on_event_25() { y++ }  
  
on_upload_start() { allocate buffer }  
on_upload_complete() { use & free buffer }
```

What if we receive multiple upload start messages? Or complete messages?

Easy to spot in small examples. Hard in large drivers. 100 events. Large code.

No FSM Trap - User Interfaces

Despite many state machine libraries, not very common yet.

Similar evolution problems.

UI doesn't need state machine at start.

When statefulness is needed, **spaghetti variables** are used.



State
Machines

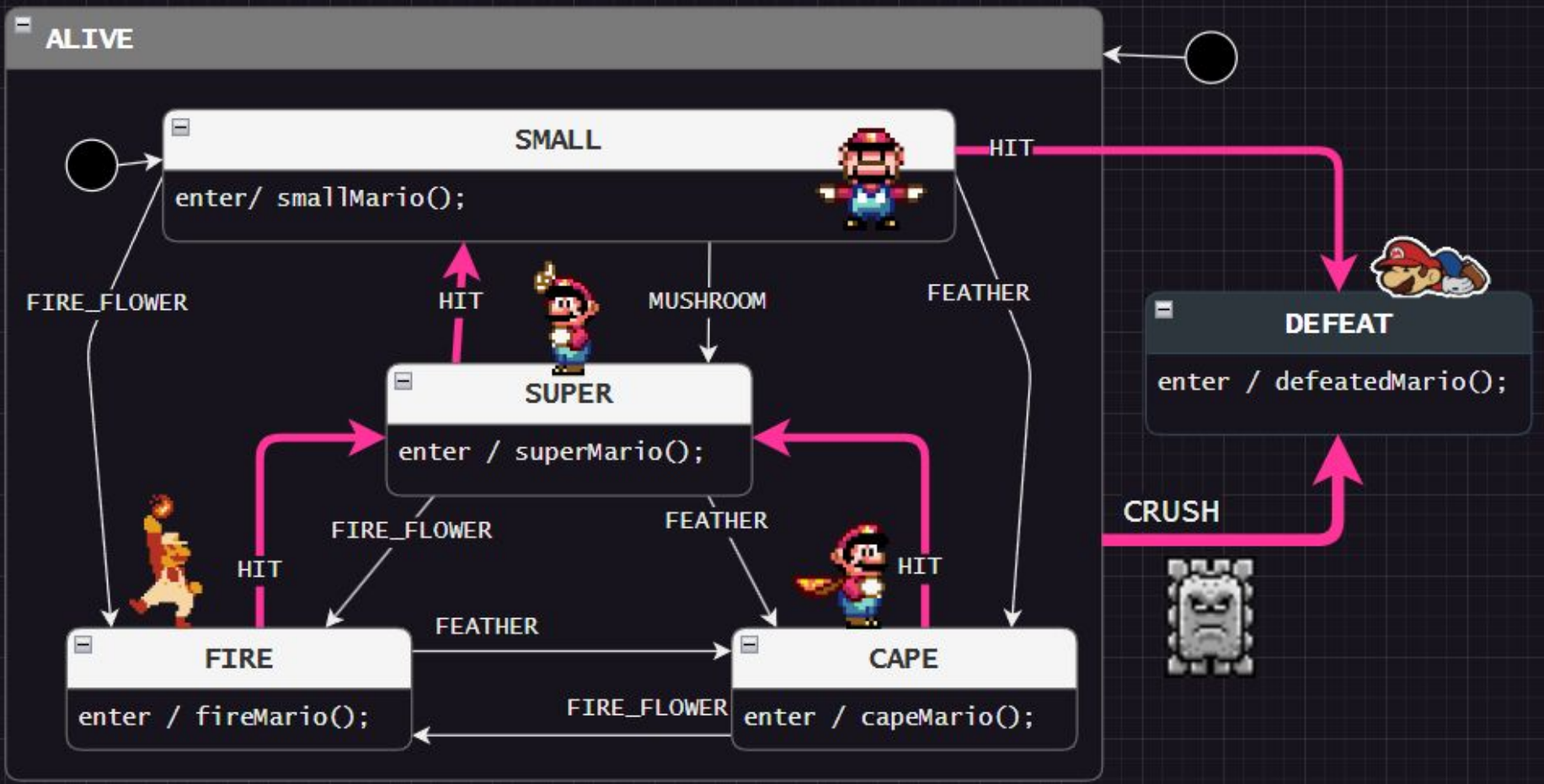


Spaghetti
Variables

Spaghetti Variables For Statefulness (not ideal)

```
class PlayerCharacter extends SomeGameObject {  
    is_on_ground = false;  
    is_crouching = false;  
    is_falling = false;  
    is_jumping = false;  
    is_attacking = false;  
    is_dead = false; //...  
  
    update() {  
        if (!is_dead && !is_attacking && !is_jumping && !is_falling) {  
            // more checks... many lines of code  
        } else {  
            // more checks... many lines of code  
        }  
    }  
}
```

StateSmith game dev example

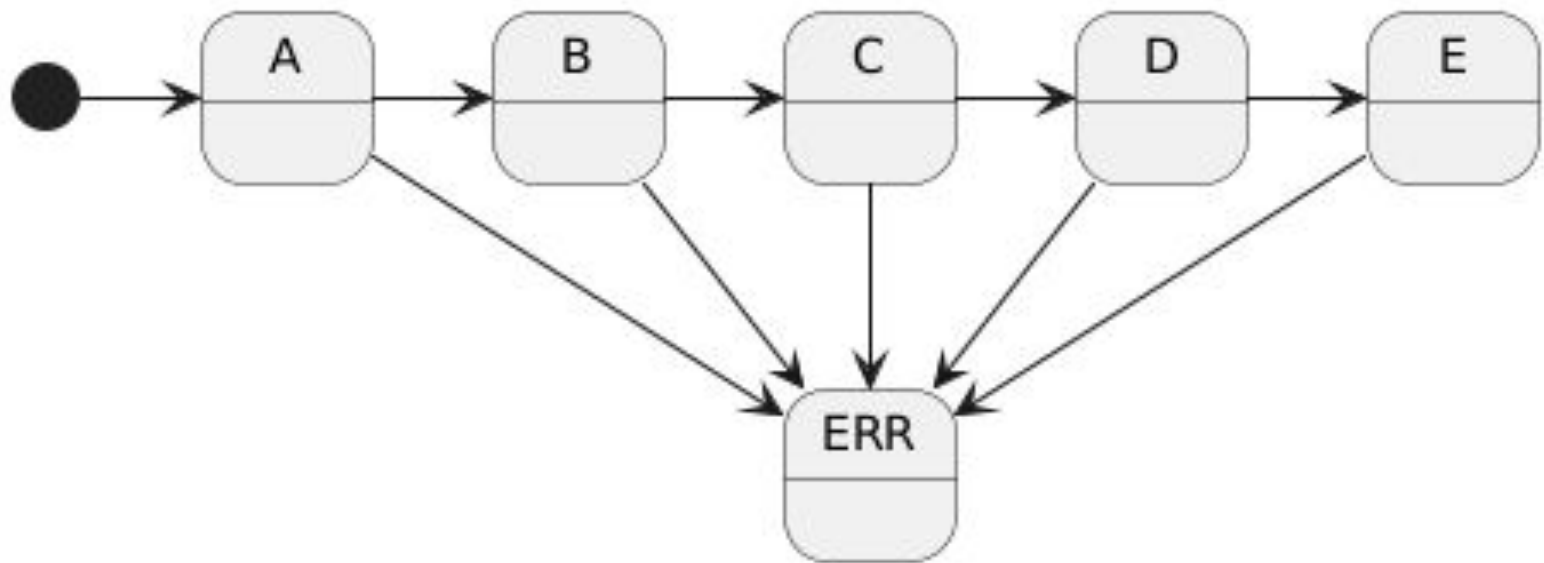


Pentest Tips

Think workflows (password reset, authentication, ordering).

Visually map the workflow. Attack each point. Try various combinations.

Look for repeated transitions for error handling. Often not fully tested by devs.



Hand Coded State Machines Aren't Perfect

Small hand coded state machines are easy to implement.

As designs grow, hand coded designs can become a nightmare.

2000+ line switch statements become **their own security vulnerability**.

```
void state_machine(StateMachine *sm, Event event) {  
    switch (sm->current_state) {  
        case STATE_A:  
            if (event == EVENT_NEXT) {  
                do_x();  
                sm->current_state = STATE_B;  
            }  
            break;  
    }  
}
```

StateSmith Helps Create Secure Maintainable code

Generate code from the diagram! This is really powerful.

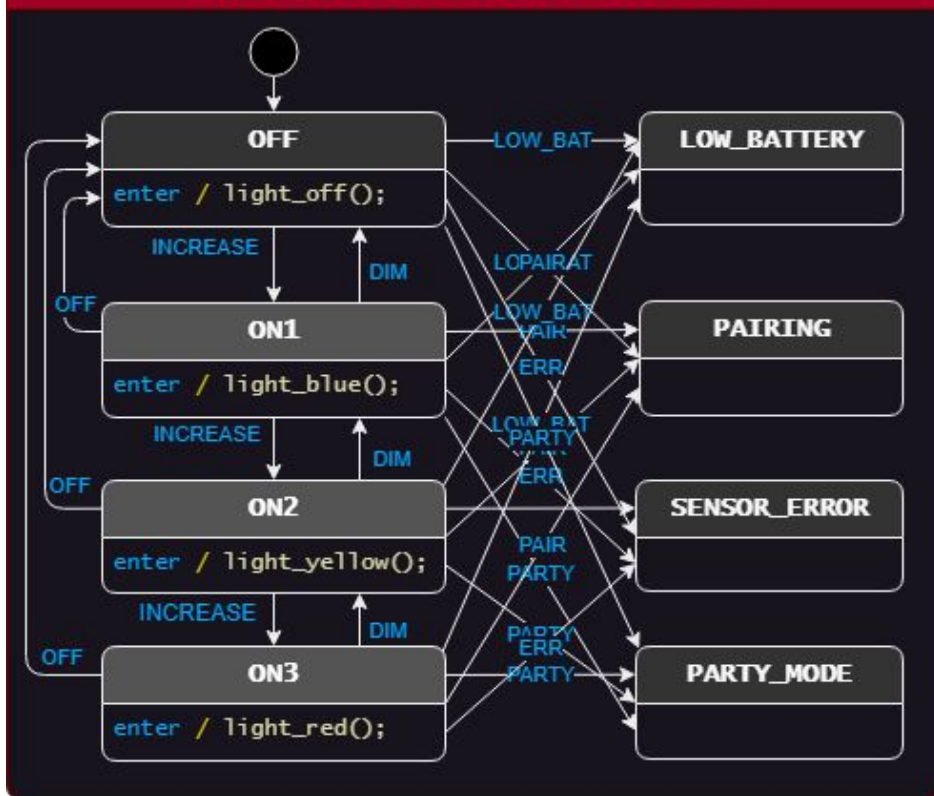
Manually creating diagrams of your code isn't sustainable. They rot & lie.

Hierarchical State Machines are powerful.

Enter and Exit behaviors are great for handling resources and preventing bugs.

Hierarchical State Machines vs Flat State Machines

REGULAR FLAT FINITE STATE MACHINE



HIERARCHICAL STATE MACHINE

