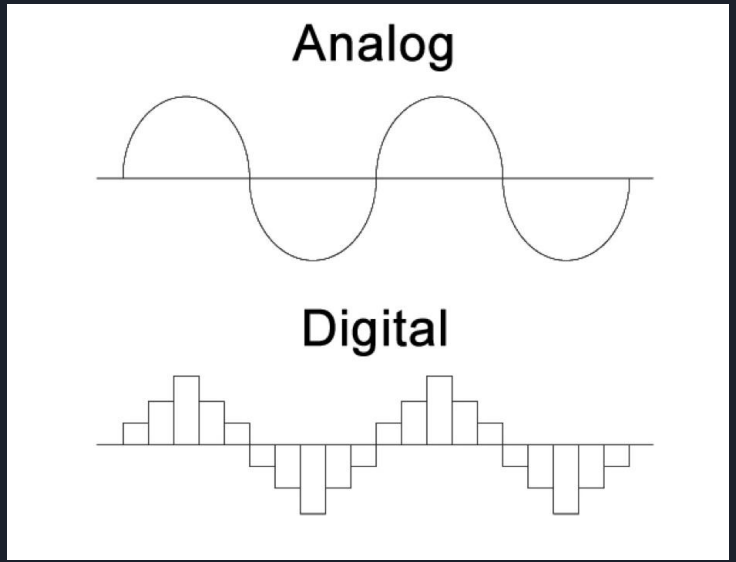
Hardware Hacking

bottom text

Digital electronics are a lie

- Ok, not really
  - but computers really just work by detecting two states
  - (voltage or no voltage)
  - Each chip has its own “cutoff voltage”
    - < counted as a 0
    - > counted as a 1
    - In between is undefined



Hardware in the realm of security

- IoT
- Embedded Systems
- Game Consoles
- Personal Computers

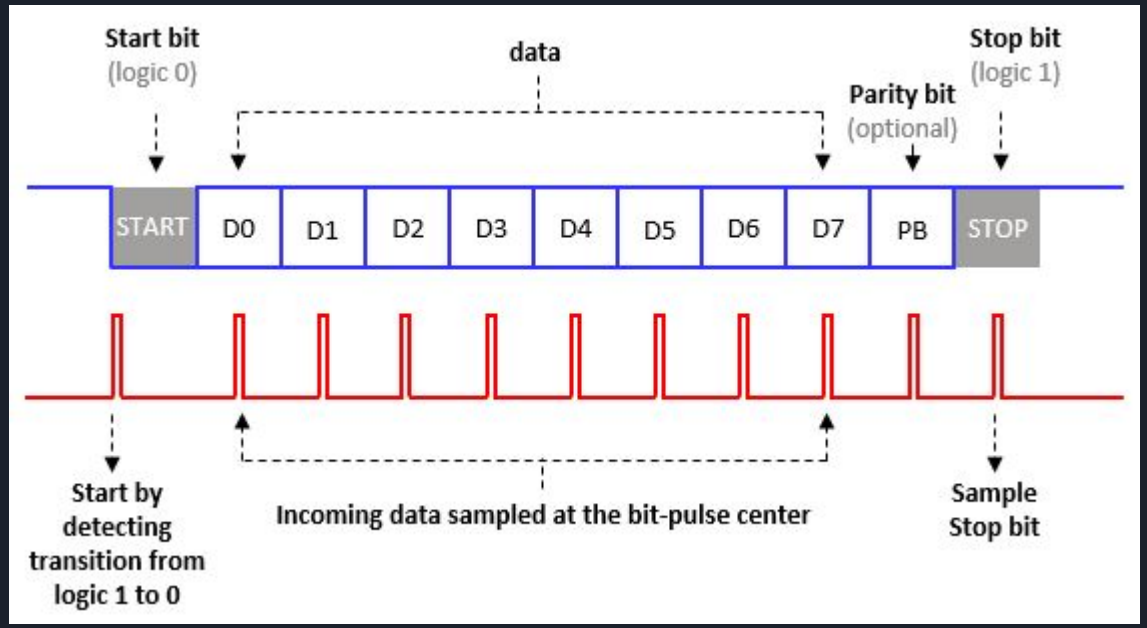
Classes of Hardware Exploits

- Code execution as a feature
  - Exposed Debug Interfaces
- Glitching attacks
- Memory attacks

Debug/Communication Interfaces

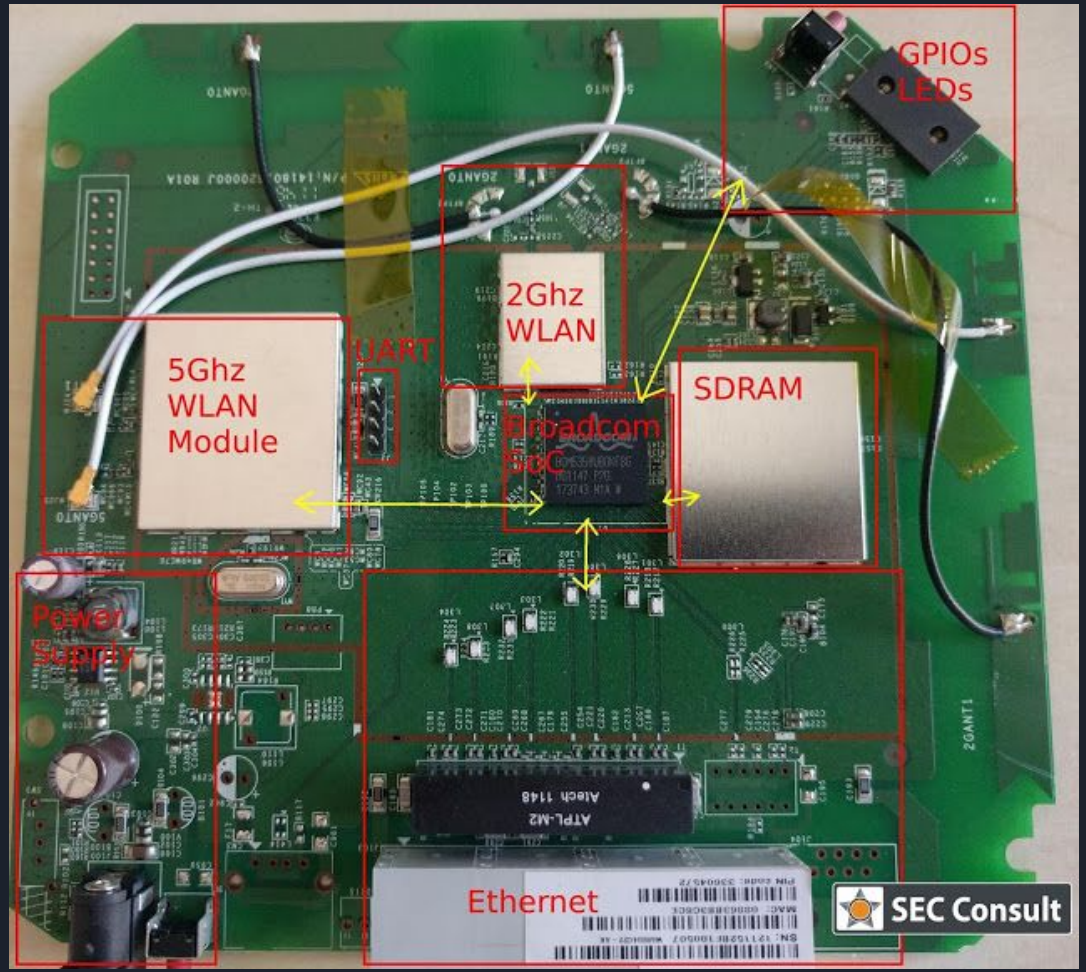
- Components have to communicate with each other
- Examples of debug/communication protocols
  - JTAG (Joint Test Action Group)- debug
  - UART (Universal Asynchronous Receiver Transmitter)- communication or debug (no_clk)
  - I2C - communication (clk)
  - SPI (Serial Peripheral Interface) - communication (clk)
  - DMA (direct memory access) - communication (clk)

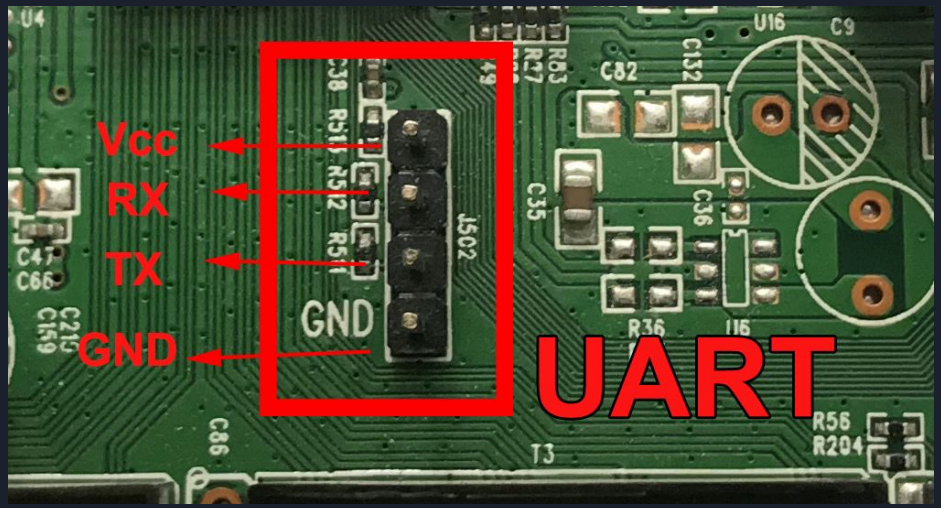
UART

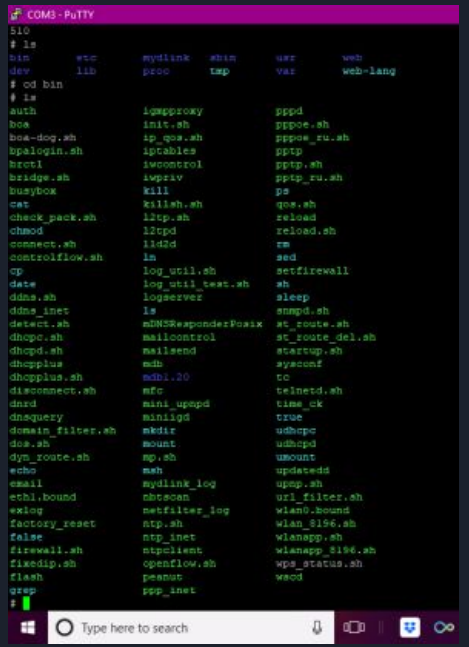

Start bit
(logic 0)
data
Parity bit
(optional)
Stop bit
(logic 1)
START
D0
D1
D2
D3
D4
D5
D6
D7
PB
STOP
Start by detecting transition from logic 1 to 0
Incoming data sampled at the bit-pulse center
Sample Stop bit

Debug/Communication Interfaces

- Usually, embedded systems have hardware support for these protocols
  - All the programmer has to do is poll a register or have an interrupt occur to rx/tx data
- This stuff is usually exposed on lower-end embedded devices like routers
- Lots of IoT devices run Linux where UART is configured as a console
  - Result: usually free root


GPIOs
LEDs
2Ghz
WLAN
5Ghz
WLAN
Module
UART
Broadcom
SoC
SDRAM
Power
Supply
Ethernet
SEC Consult


Vcc
RX
TX
GND
GND
UART
U4
U16
C9
C82
C132
C35
C36
C47
C66
C225
C159
C86
R36
I16
R56
R204
T3
R83
R27
R511
R512
R511
JS02


COM3 - PuTTY
510
ls
bin etc mydlink sbin usr web
dev lib proc tmp var web-lang
cd bin
ls
auth igmpproxy pppd
boa init.sh pppoe.sh
boa-dog.sh ip_qos.sh pppoe_ru.sh
bpalogin.sh iptables pptp
brctl iwcontrol pptp.sh
bridge.sh iwpriv pptp_ru.sh
busybox kill ps
cat killsh.sh qos.sh
check_pack.sh l2tp.sh reload
chmod l2tpd reload.sh
connect.sh lld2d rm
controlflow.sh ln sed
cp log_util.sh setfirewall
date log_util_test.sh sh
ddns.sh logserver sleep
ddns_inet ls snmpd.sh
detect.sh mDNSResponderPosix st_route.sh
dhcpc.sh mailcontrol st_route_del.sh
dhcpd.sh mailsend startup.sh
dhcpplus mdb sysconf
dhcpplus.sh mdb1.20 tc
disconnect.sh mfc telnetd.sh
dnrd mini_upnpd time_ck
dnsquery miniigd true
domain_filter.sh mkdir udhcpc
dos.sh mount udhcpd
dyn_route.sh mp.sh umount
echo msh updatedd
email mydlink_log upnp.sh
eth1.bound nbtscan url_filter.sh
exlog netfilter_log wlan0.bound
factory_reset ntp.sh wlan_8196.sh
false ntp_inet wlanapp.sh
firewall.sh ntpclient wlanapp_8196.sh
fixedip.sh openflow.sh wps_status.sh
flash peanut wsod
grep ppp_inet

Type here to search

Glitching Attacks

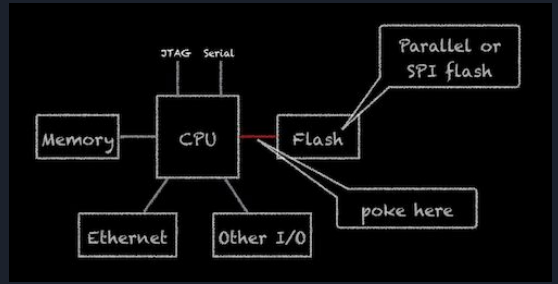
- Abuse the fact that ‘digital electronics don’t exist’
- Change the voltage of some hardware so that the software gets confused
- Example: Xbox 360 RGH/JTAG, Wii TWIIZERS, IoT PIN2PWN

Circuits

- For basic resistive circuits: V=IR
  - Active circuits are more complicated but not super important for this talk
- What happens if you short two lines together?
  - They become parallel which means that their voltages are the same
- Idea: short two pins on a flash chip together, causing read errors and undefined behavior
  - Pin2Pwn :)

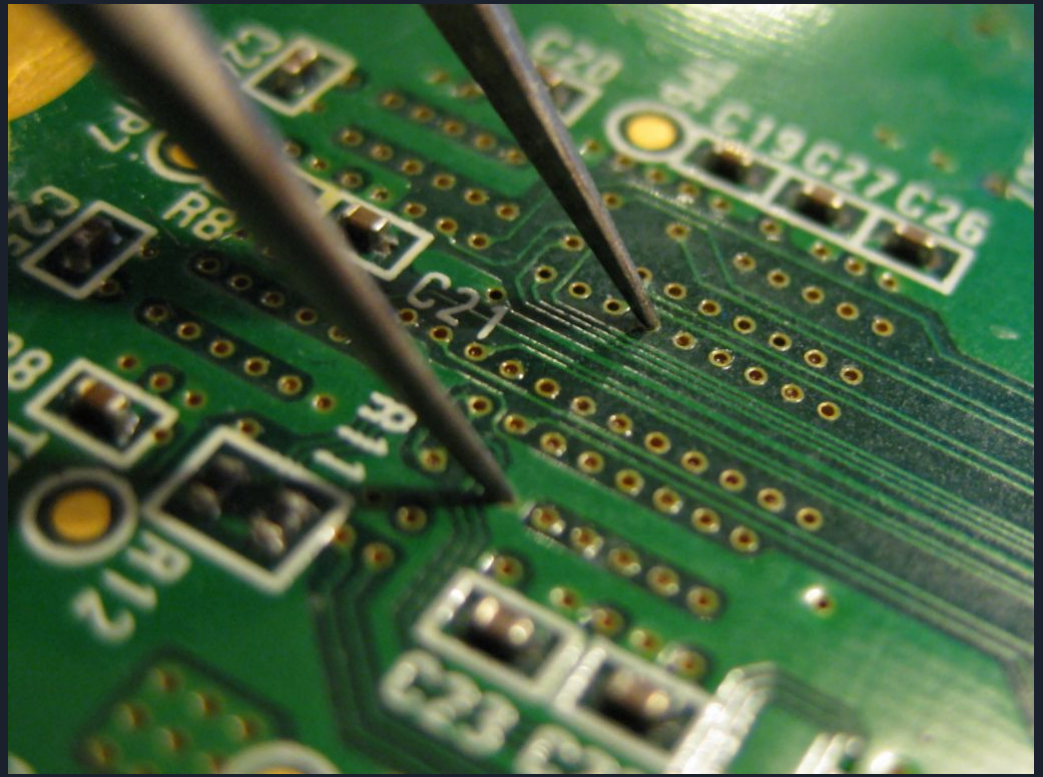
PIN2PWN

- During the boot process, some devices don't have a pre-set failure path
- If you disrupt the flash chip's communication to the CPU, you might get a bootloader shell
  - Why? Bootloader trys to pull the kernel from flash and fails, then panics
  - Many devices weren't built with this in mind and the default behavior of U-Boot is to give you a shell
- Why does this matter?
  - Many manufactures disable UART or JTAG as a security measure by cutting the traces pre-manufacturing OR disabling them in software
  - But using this method, you are usually able to interrupt the bootloader and if the interfaces are only disabled in software, you can re-enable them using a kernel argument (for linux devices only)



RGH

Twiizers



Memory Attacks

- If you have access to system memory, game over
- You can just patch the kernel to disable passwords or do anything you want really
  - (this is still mostly true, however OS X and Windows have some increased protection for authentication with encryption/virtualization)

But how do you write to system memory without access to the OS?

DMA

- Remember DMA?
- Turns out, a lot of peripherals actually require/allow DMA to function
  - Namely PCI Express
- PCI express on laptops?
  - THUNDERBOLT!!!!
  - If you are a recognized device or can bypass the security mode for Thunderbolt, you can write/read to memory at will
  - Turns out you can actually write the thunderbolt firmware with a simple SPI programmer to patch out the security checks (this attack is called Thunderspy)

THUNDERSPY

When Lightning Strikes Thrice: Breaking Thunderbolt 3 Security

By Björn Ruytenberg

Thunderspy targets devices with a Thunderbolt port. If your computer has such a port, an attacker who gets brief physical access to it can read and copy all your data, even if your drive is encrypted and your computer is locked or set to sleep.

Thunderspy is stealth, meaning that you cannot find any traces of the attack. It does not require your involvement, i.e., there is no phishing link or malicious piece of hardware that the attacker tricks you into using. Thunderspy works even if you follow best security practices by locking or suspending your computer when leaving briefly, and if your system administrator has set up the device with Secure Boot, strong BIOS and operating system account passwords, and enabled full disk encryption. All the attacker needs is 5 minutes alone with the computer, a screwdriver, and some easily portable hardware.

We have found 7 vulnerabilities in Intel's design and developed 9 realistic scenarios how these could be exploited by a malicious entity to get access to your system, past the defenses that Intel had set up for your protection.

We have developed a free and open-source tool, Spycheck, to determine if your system is vulnerable. If it is found to be vulnerable, Spycheck will guide you to recommendations on how to help protect your system.

Update 14 Aug 2020
The slides from our Black Hat USA talk are online here.

Update 6 Aug 2020
Today, we are happy to release two tools to patch your system to achieve the same (partial) security that Intel's Kernel DMA Protection offers against Thunderspy. These tools aim to bring Kernel DMA Protection to all systems released between 2013 and 2020 that do not ship Kernel DMA Protection, but are in fact technically capable. Please visit our Thunderspy 2 page for more details. The same fix could be provided systematically by vendors if they update their UEFI distributions.

Vulnerability report | Cite | Spycheck for Windows | Spycheck for Linux

Spycheck supports Windows 7, 8.x and all builds of 10, as well as Linux kernel 3.16 and Python 3.4, and later. Source code on GitHub.
Prefer checking manually? Scroll down to the instructions for Windows, Linux and MacOS.

Questions?