Please turn off WiFi and switch to flight mode!

Jailbreaking the Apple HomePod

Fun with checkm8 & smart speakers

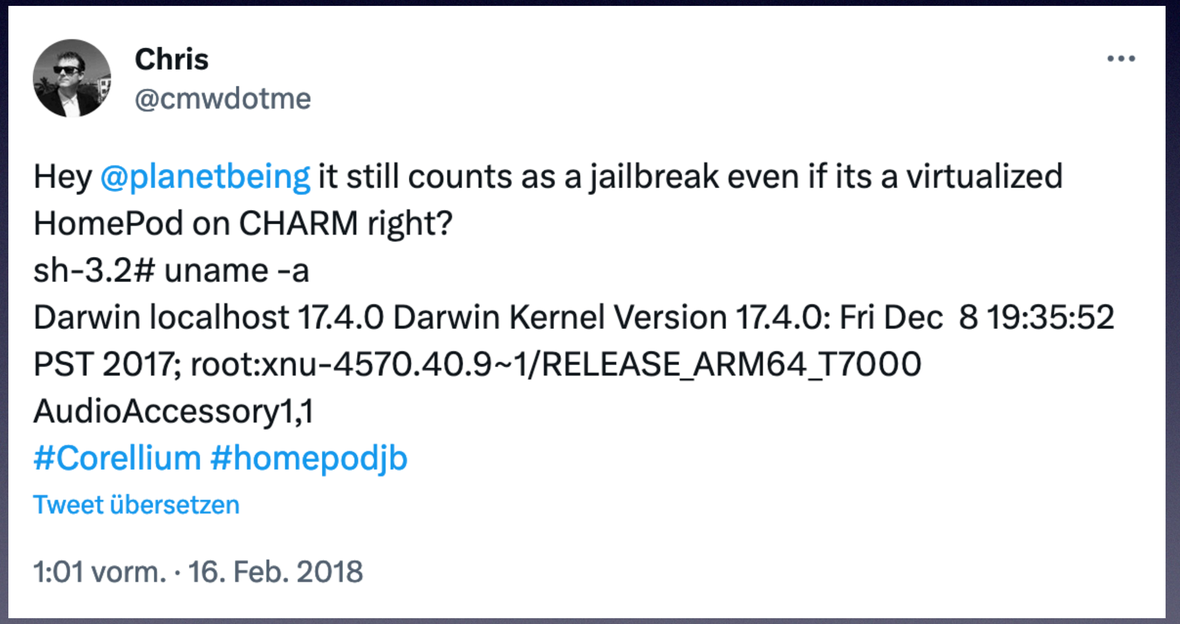
Target

- HomePod is Apple's smart speaker (similar to Amazon's Alexa)
- Used for Music & smart home controlling
- Released in October 2020
- Runs a custom down version of ~~iOS~~ tvOS
- A8 CPU (same as iPhone 6)
  - Vulnerable to checkm8 BootROM exploit

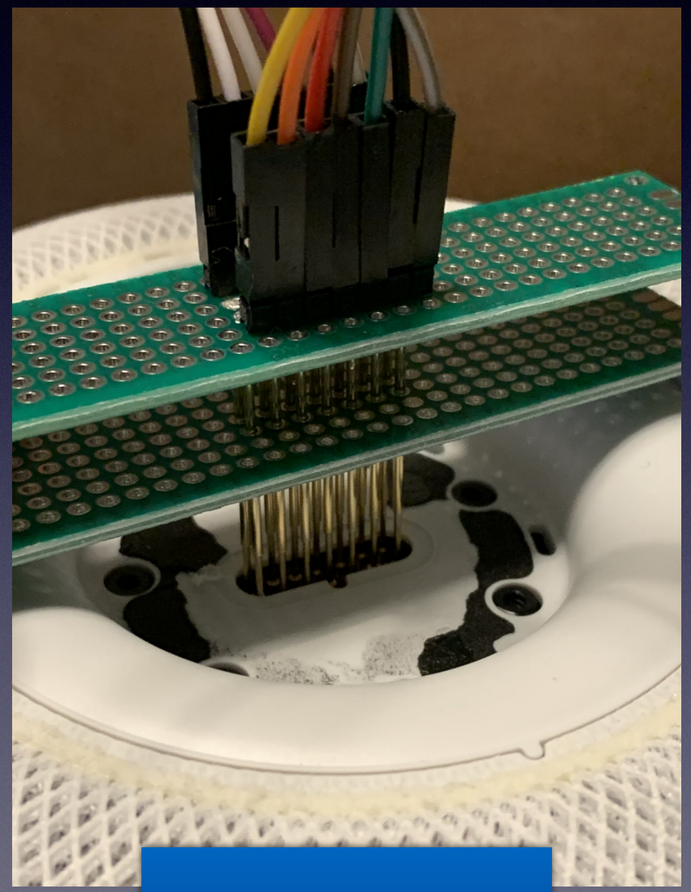


Previous work

- HomePod OS/kernel virtualized by Corellium
- Not a real jailbreak
- Doesn't count :P



Previous work: @el1ng0

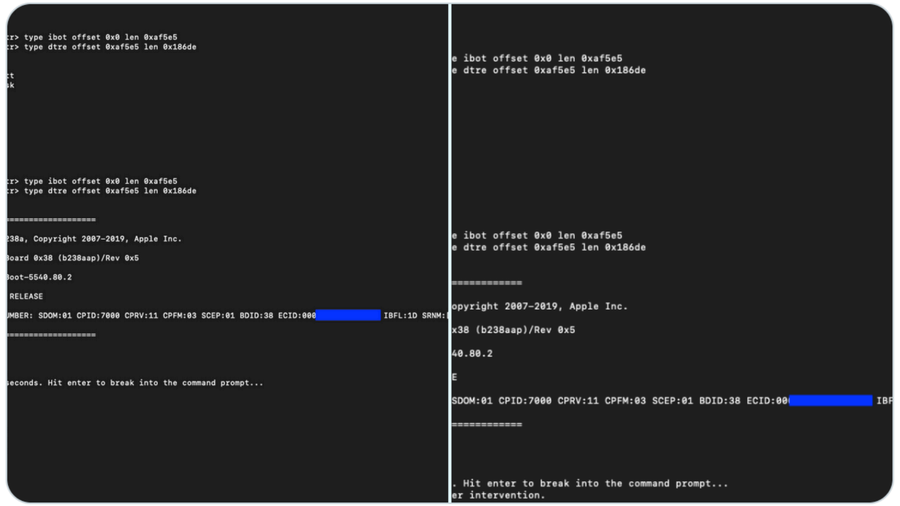


Connector reversing

L1ng
@el1ng0

Hello Homepod UART console. No special adapters needed. Next up -- let's get this DFU-ed and checkm8-ed.

Tweet übersetzen



3:10 vorm. · 21. Feb. 2020

UART + USB

L1ng
@el1ng0 Folgt Dir

Seit November 2019 bei Twitter

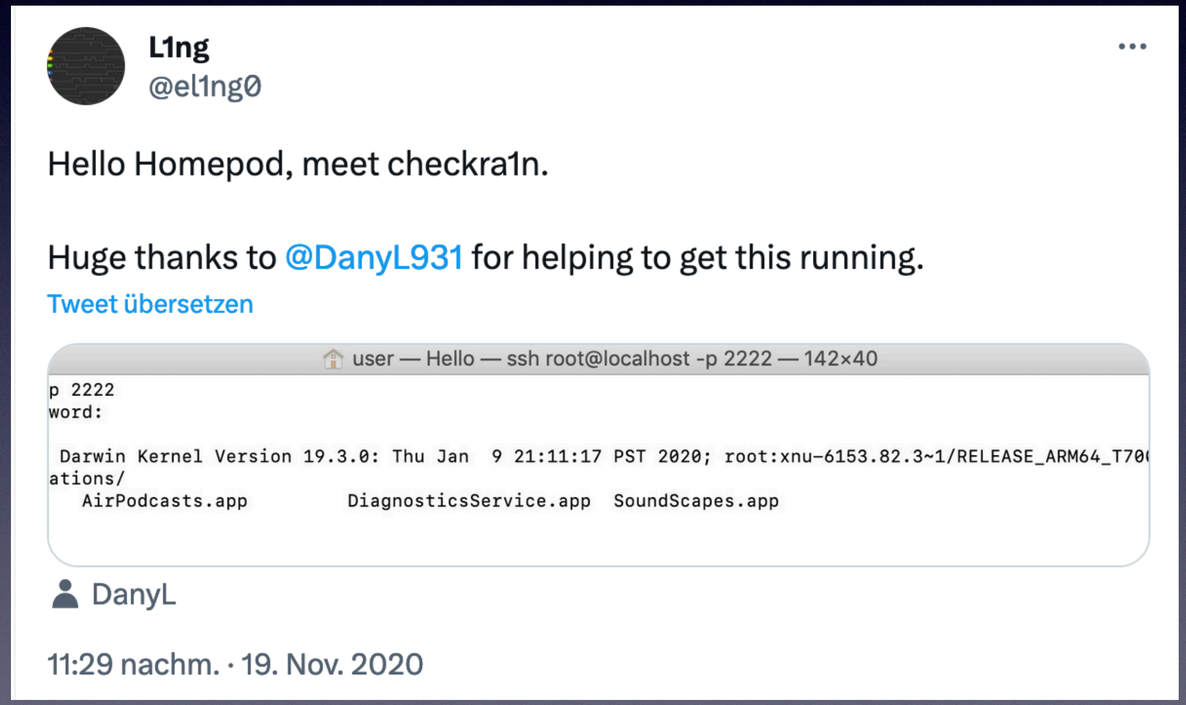
154 Folge ich **1.286** Follower



Custom connector PCB design

Previous Work

- Checkra1n + PongoOS running on HomePod
- SSH working
- So, what is left to do???



Current status

- HomePod jailbreak never became a thing
- Official checkra1n stopped working with iOS 15
  - Development stopped
  - Approach hit some roadblock
- HomePod USB is **not** easy plug&play
- So what does it take to jailbreak a HomePod?

Hardware

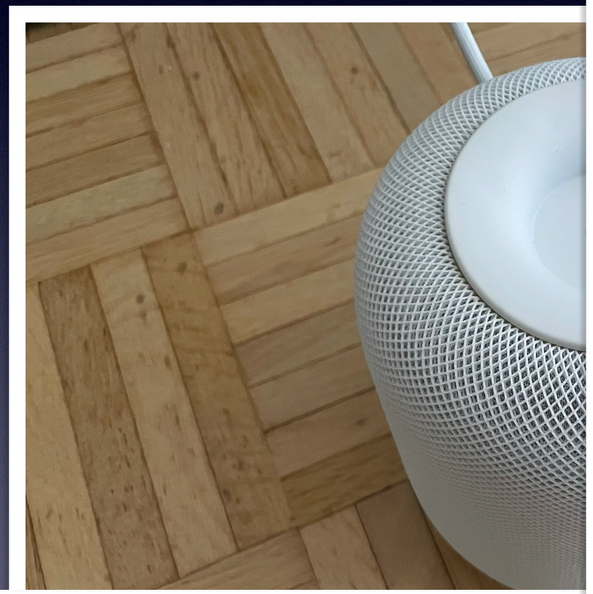
Accessing the connector

- Flip HomePod upside down
- Remove cover with heat+screwdriver
- Expose hidden port



Accessing the connector

- Flip HomePod upside down
- Remove cover with heat+screwdriver
- Expose hidden port







Accessing the connector

- Flip HomePod upside down
- Remove cover with heat+screwdriver
- Expose hidden port



Connecting to USB

- @el1ng0 published connector on github
- I couldn't figure out how to make one 🤦
- Decided to make my own cheap 3D printable connector

homepwn Public

Watch 7

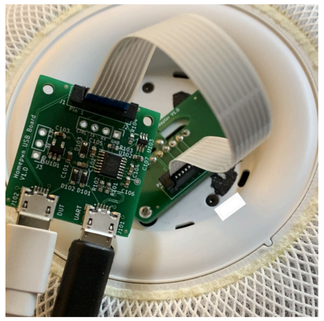
main · 1 branch · 0 tags · Go to file · Add file · Code

el1ng Update README.md — 8f6f42f on Mar 25, 2021 — 7 commits

homebreakout	Update README.md	2 years ago
homepod_usb_breakout	Update README.md	2 years ago
homepod_usb_breakout_simple	Homepwn V1.0	2 years ago
.gitignore	Initial commit	2 years ago
README.md	Update README.md	2 years ago

README.md

Homepwn

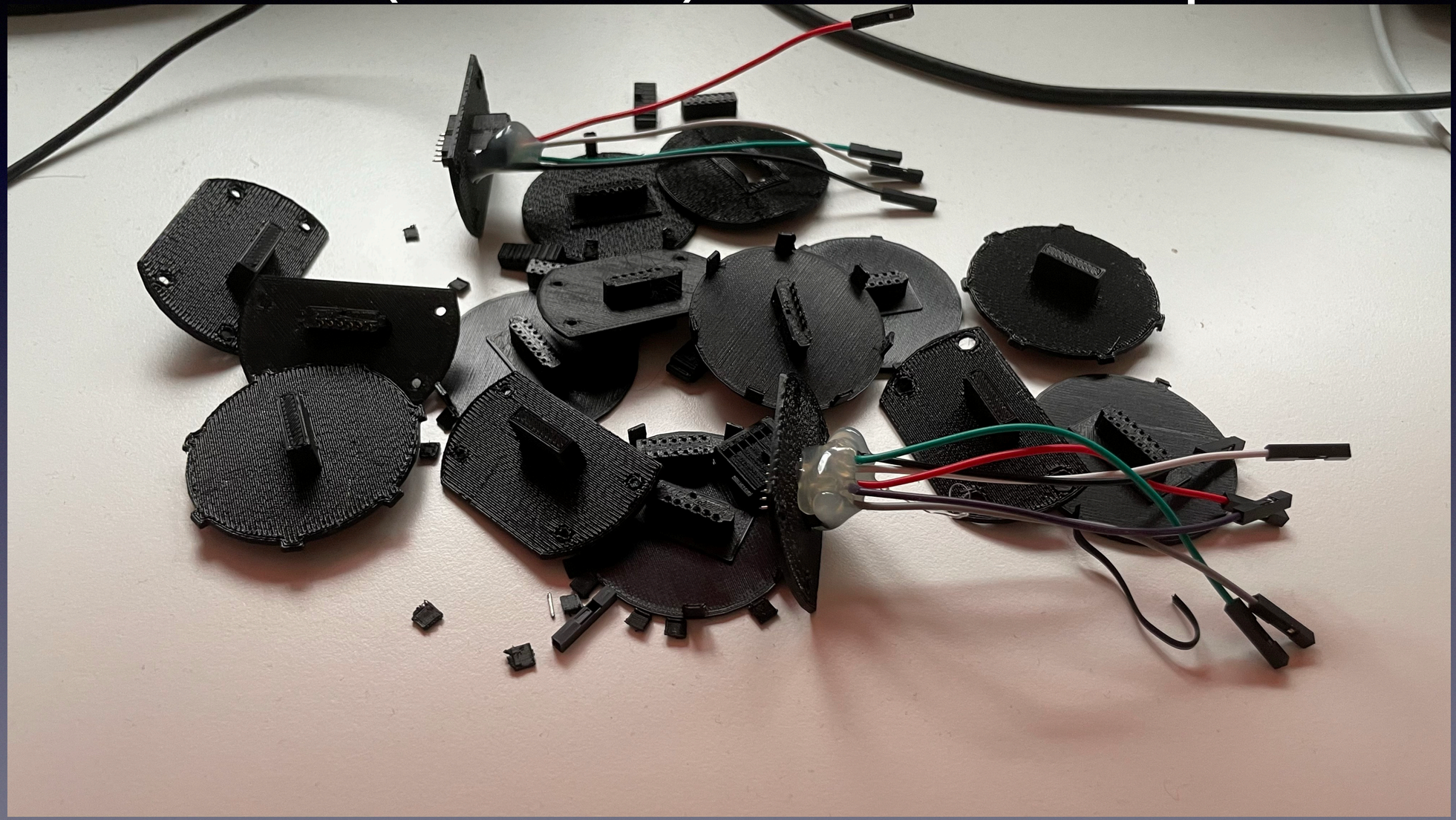


This repo contains KiCAD projects for a breakout to help jailbreak the homepod.

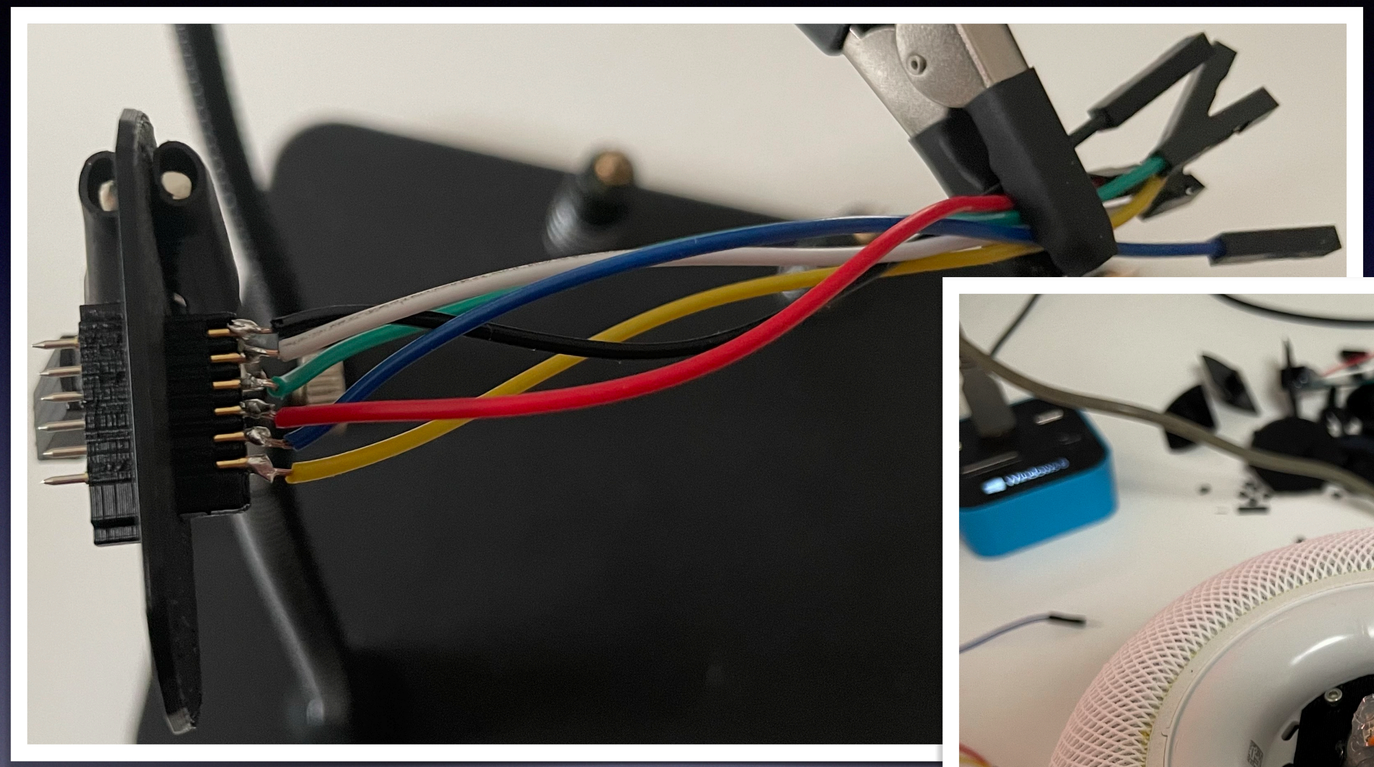
There are 3 projects:

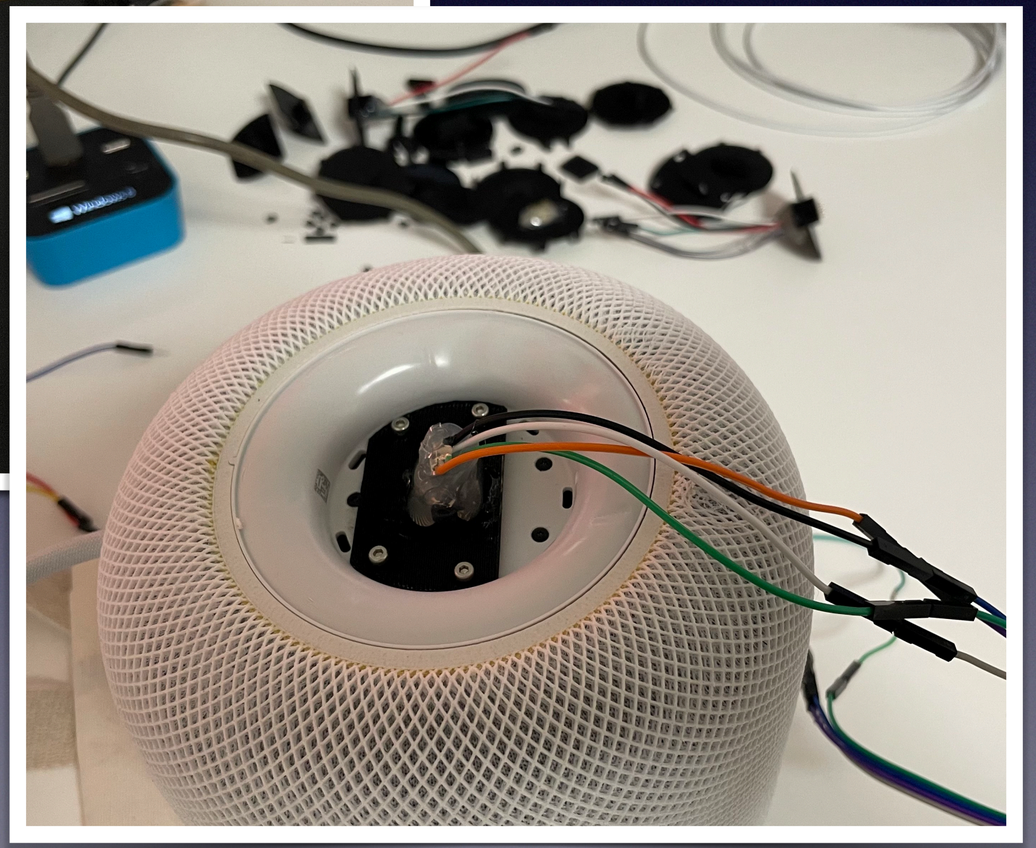
- homebreakout: This is a form fitting PCB that fits underneath the homepod and contacts the POGO pins
- homepod_usb_breakout: This is a more complex PCB that does UART to USB and provides a USB connection to the Homepod.

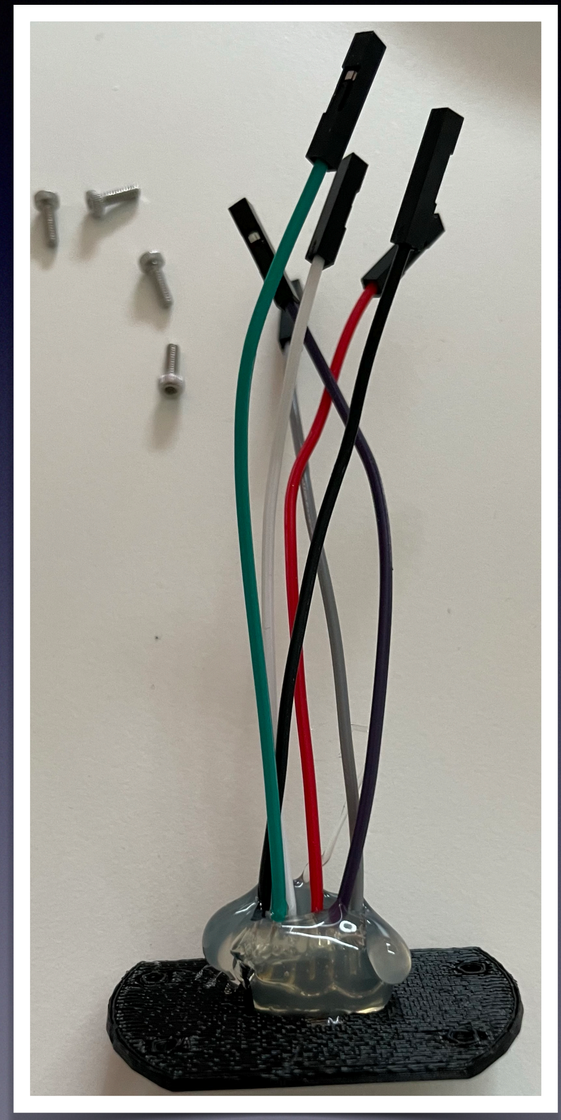
First (failed) 20 attempts



Final 3D printable connector



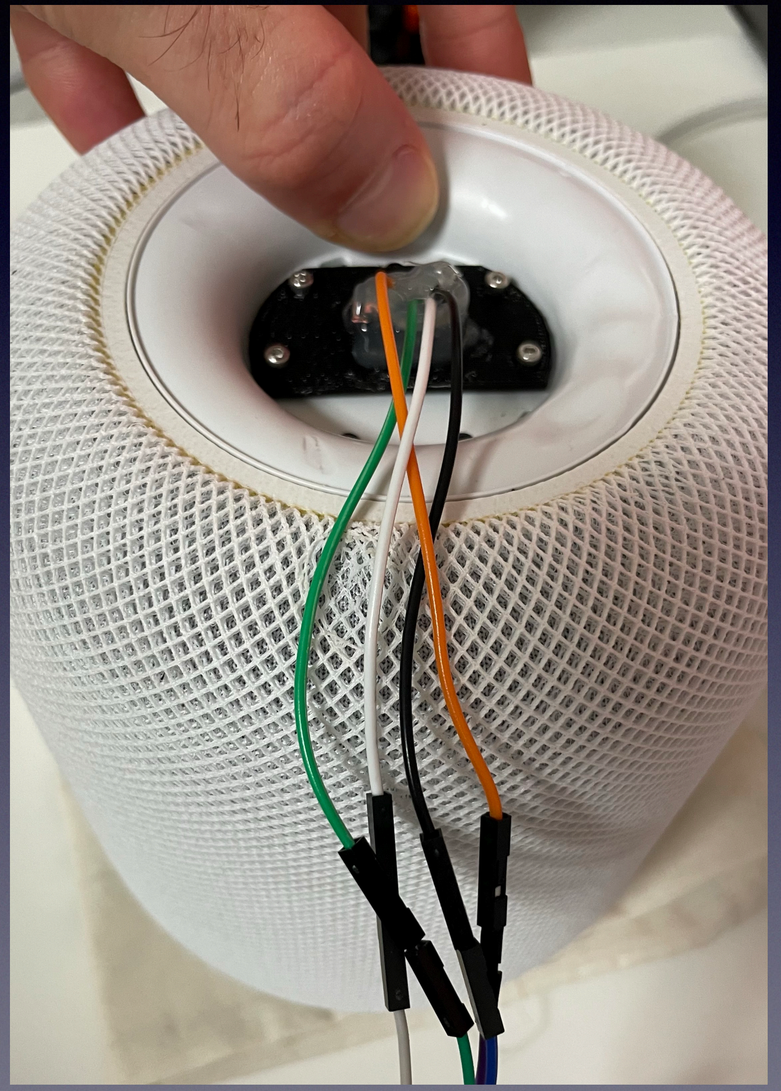




3D printable connector

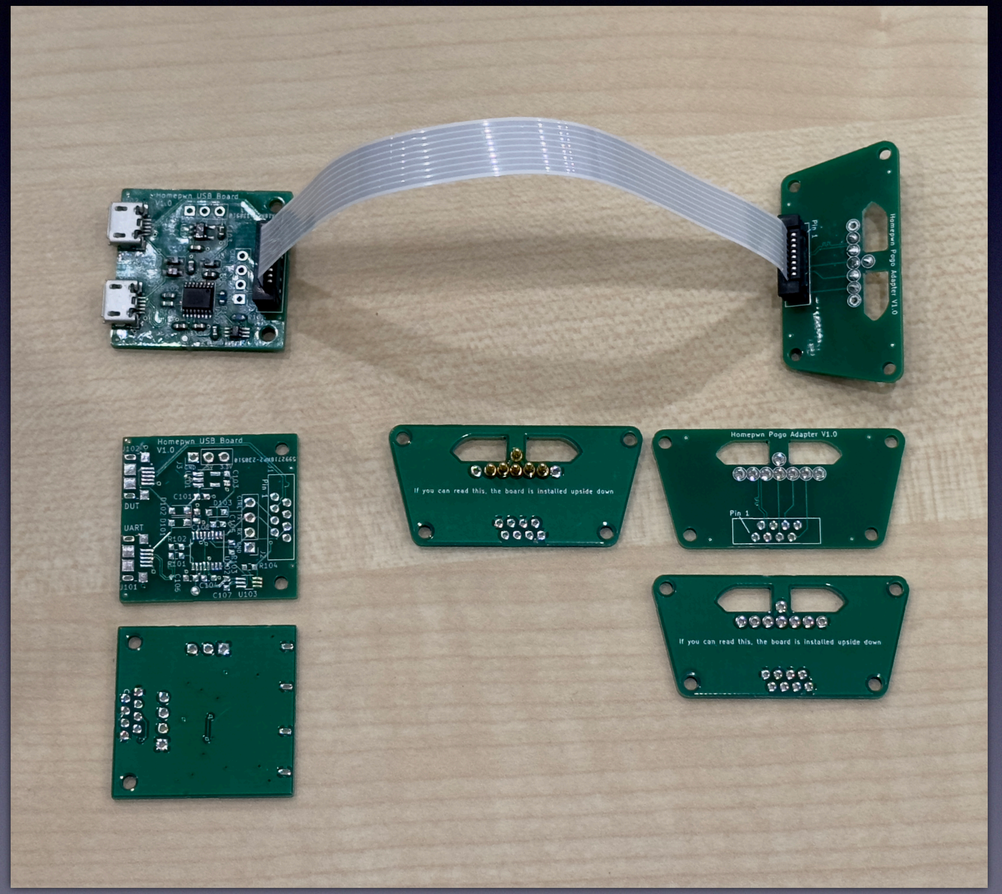
- 3D printable plastic piece
- Pogo pins 4 USB (+2 UART)
  (UART is 1.8V, not 3.3V!!!)
- Screws
- Wires
- Hot glue
- Some basic solder skills
- Total cost < 30€ for 1-25 adapters





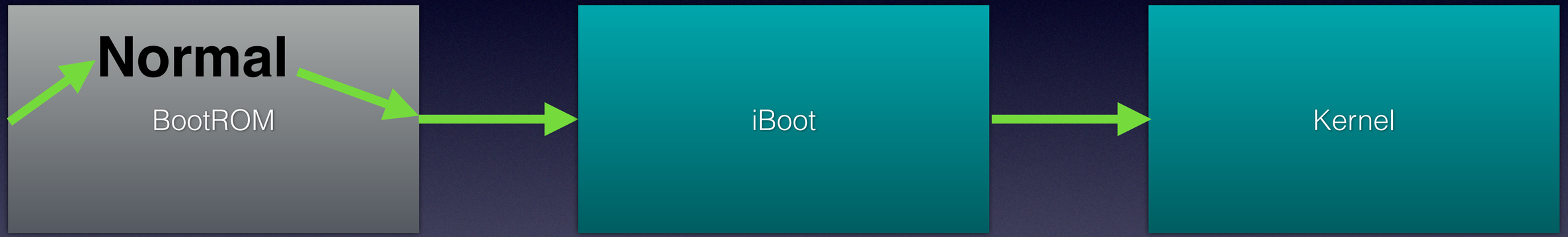
Proper connector

- Simply take the Gerber files from the repo and send them to a PCB manufacturer
  - e.g. JLCPCB, just upload the files
- Cost: About 1€ for both PCBs (in bulk), 15€ in parts (1 connector)
- Advantage: Connection is way more stable



Software

Normal iOS Boot

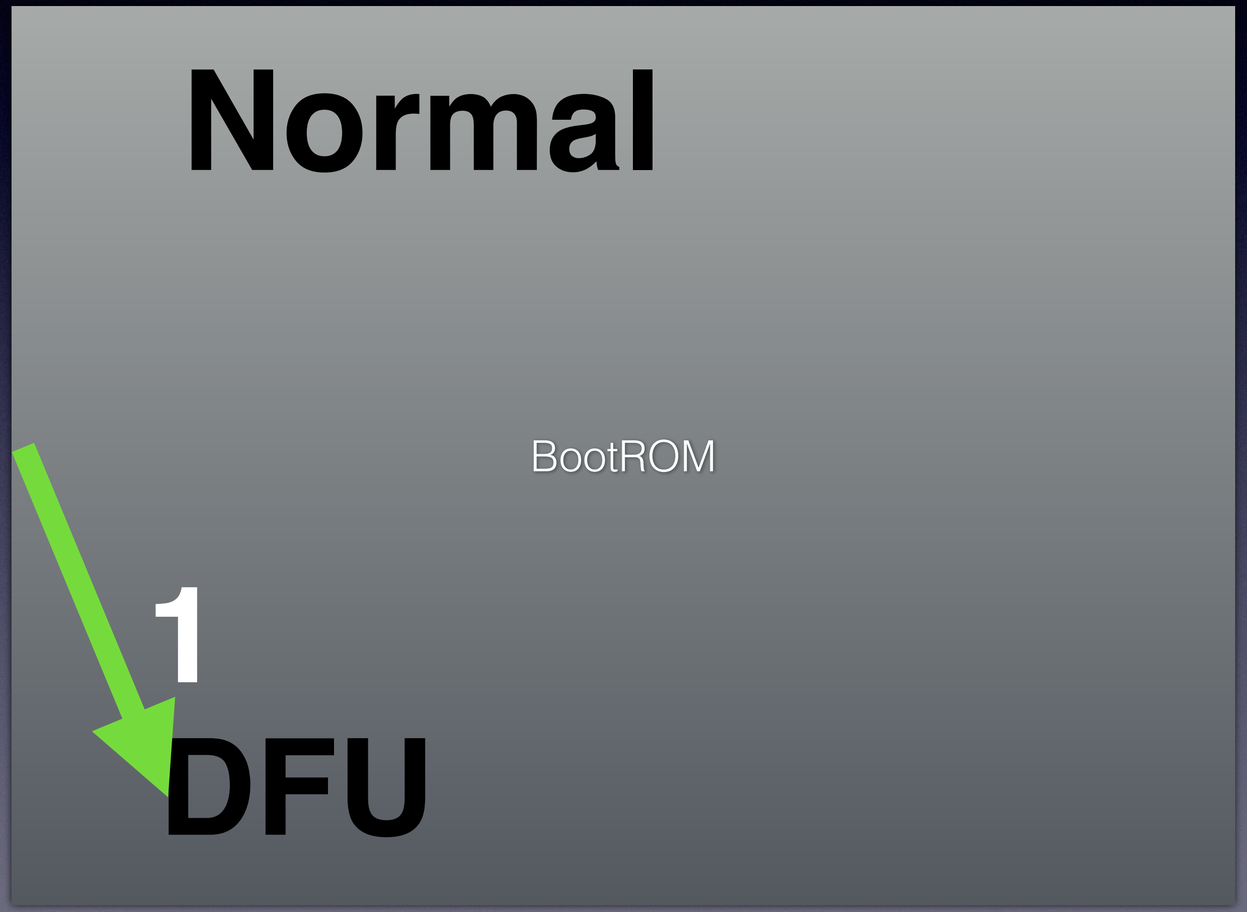


- Load iBoot from NAND
- Verify
- Boot

- Load Kernel from NAND
- Verify
- Boot

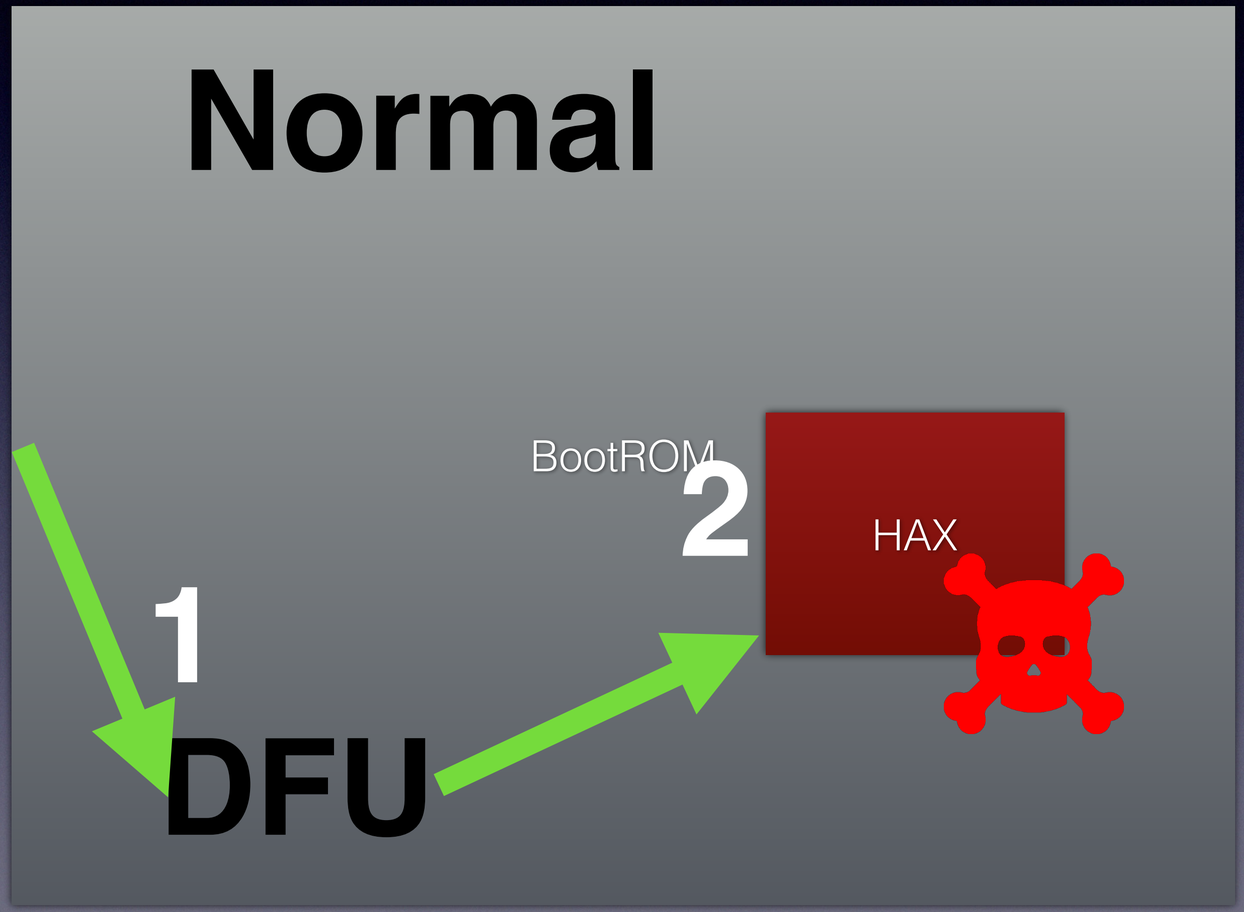
Previous BootROM exploits

- Enter DFU mode
- Exploit limra1n/SHAtter/ steaks4uce & inject Shellcode
- Let shellcode load image over USB without calling signature verification function



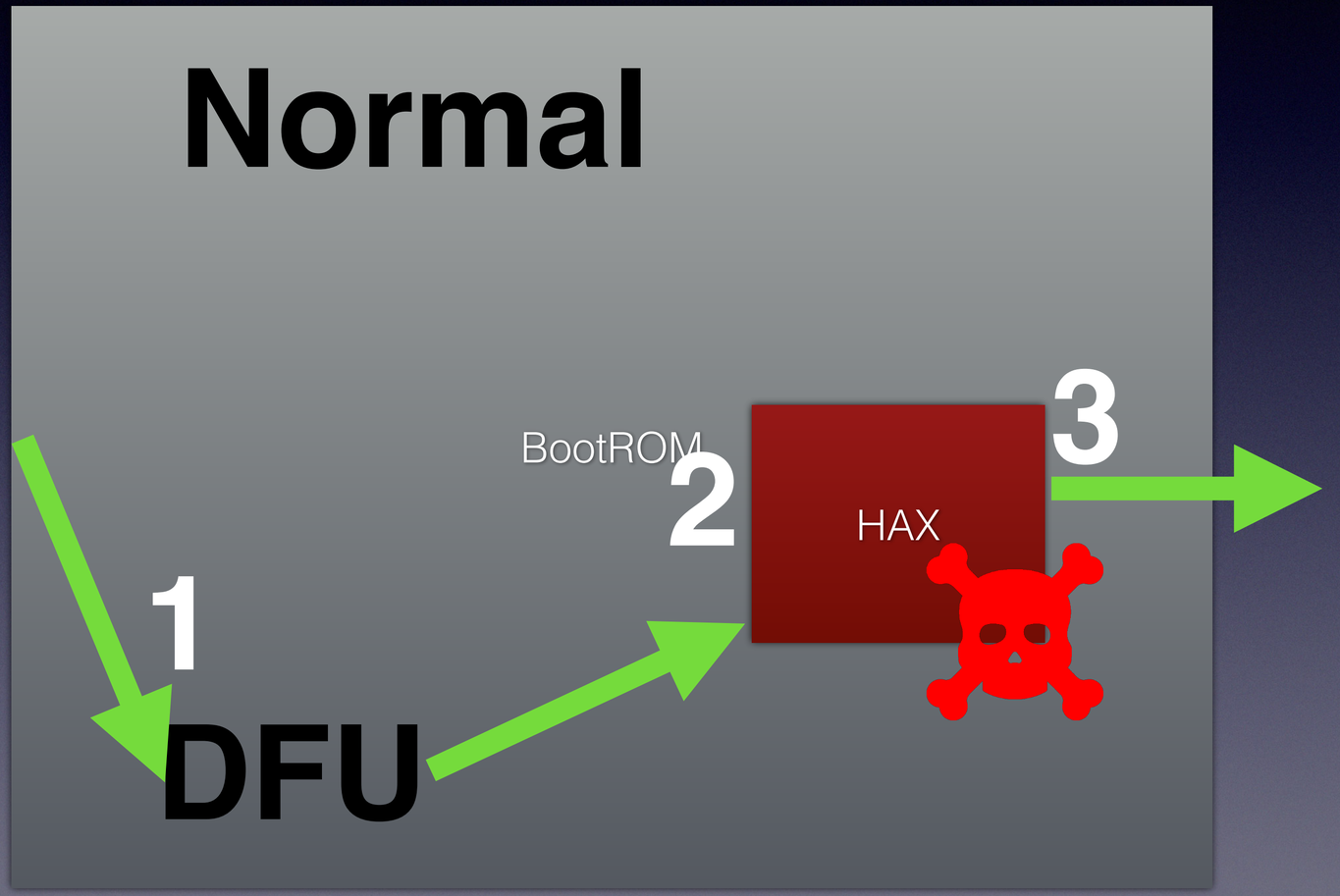
Previous BootROM exploits

- Enter DFU mode
- Exploit limra1n/SHAtter/ steaks4uce & inject Shellcode
- Let shellcode load image over USB without calling signature verification function



Previous BootROM exploits

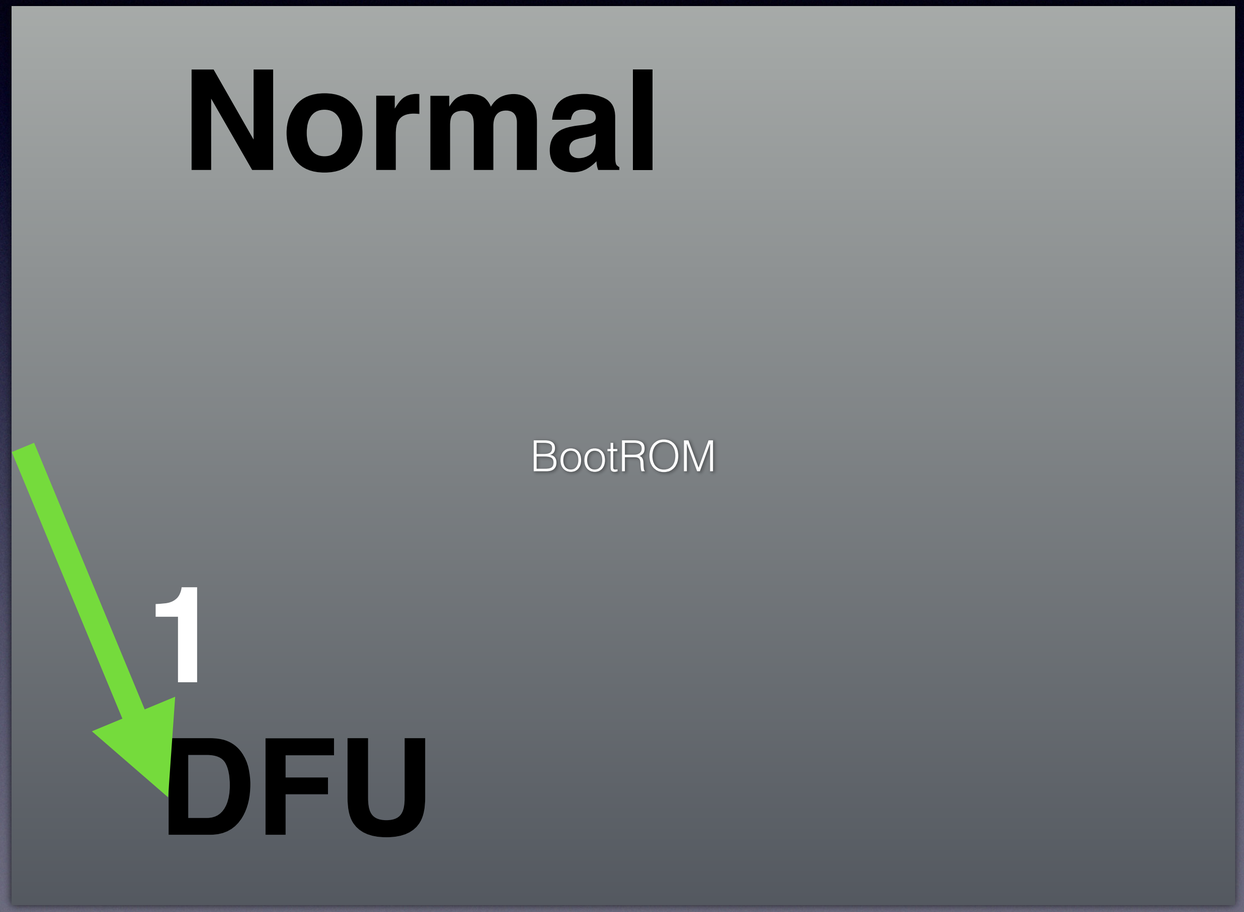
- Enter DFU mode
- Exploit limra1n/SHAtter/ steaks4uce & inject Shellcode
- Let shellcode load image over USB without calling signature verification function



Checkra1n

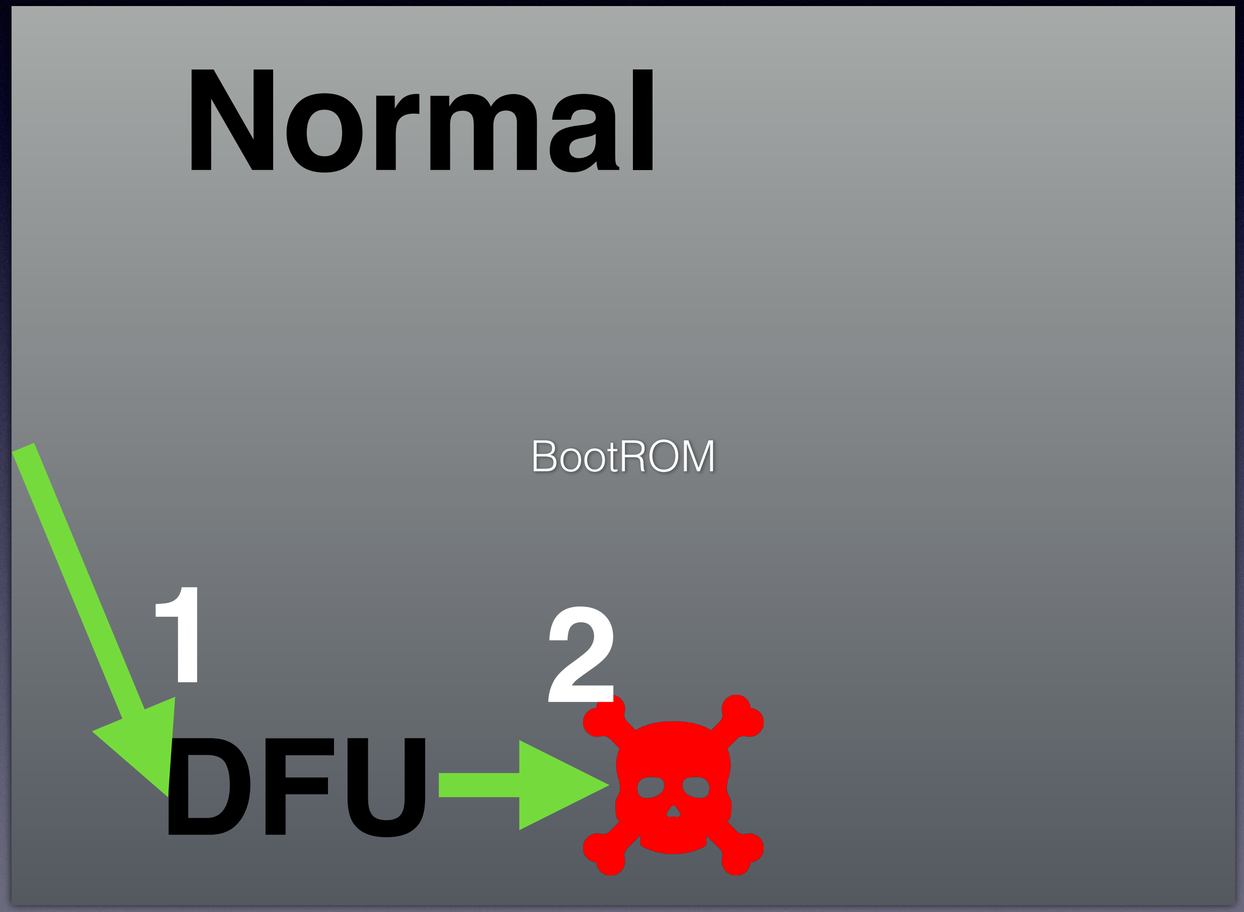
checkra1n in a Nutshell

- Enter DFU mode
- Exploit checkm8
  & inject Shellcode
- Remap BootROM page to SRAM
- Patch remapped copy



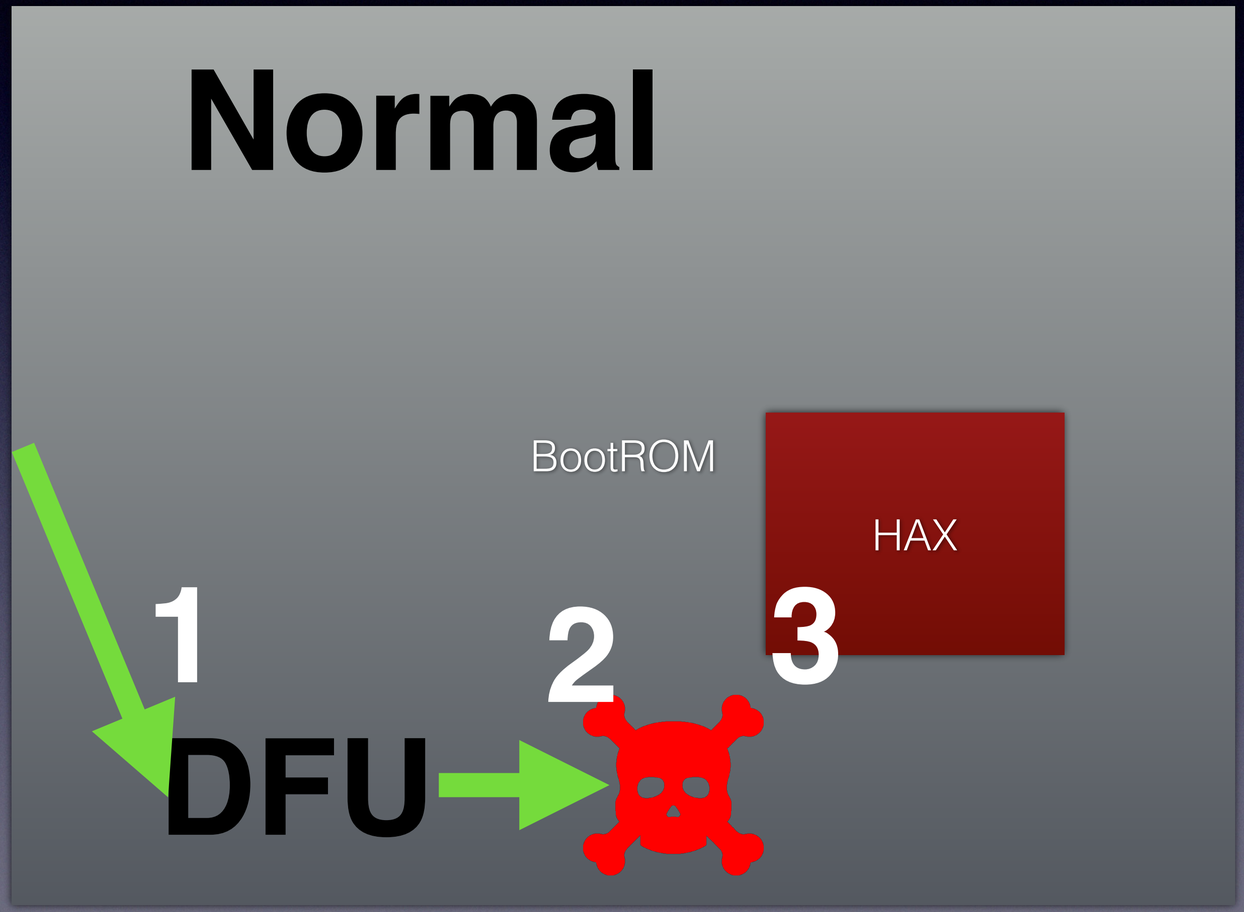
checkra1n in a Nutshell

- Enter DFU mode
- Exploit checkm8
  & inject Shellcode
- Remap BootROM page to SRAM
- Patch remapped copy



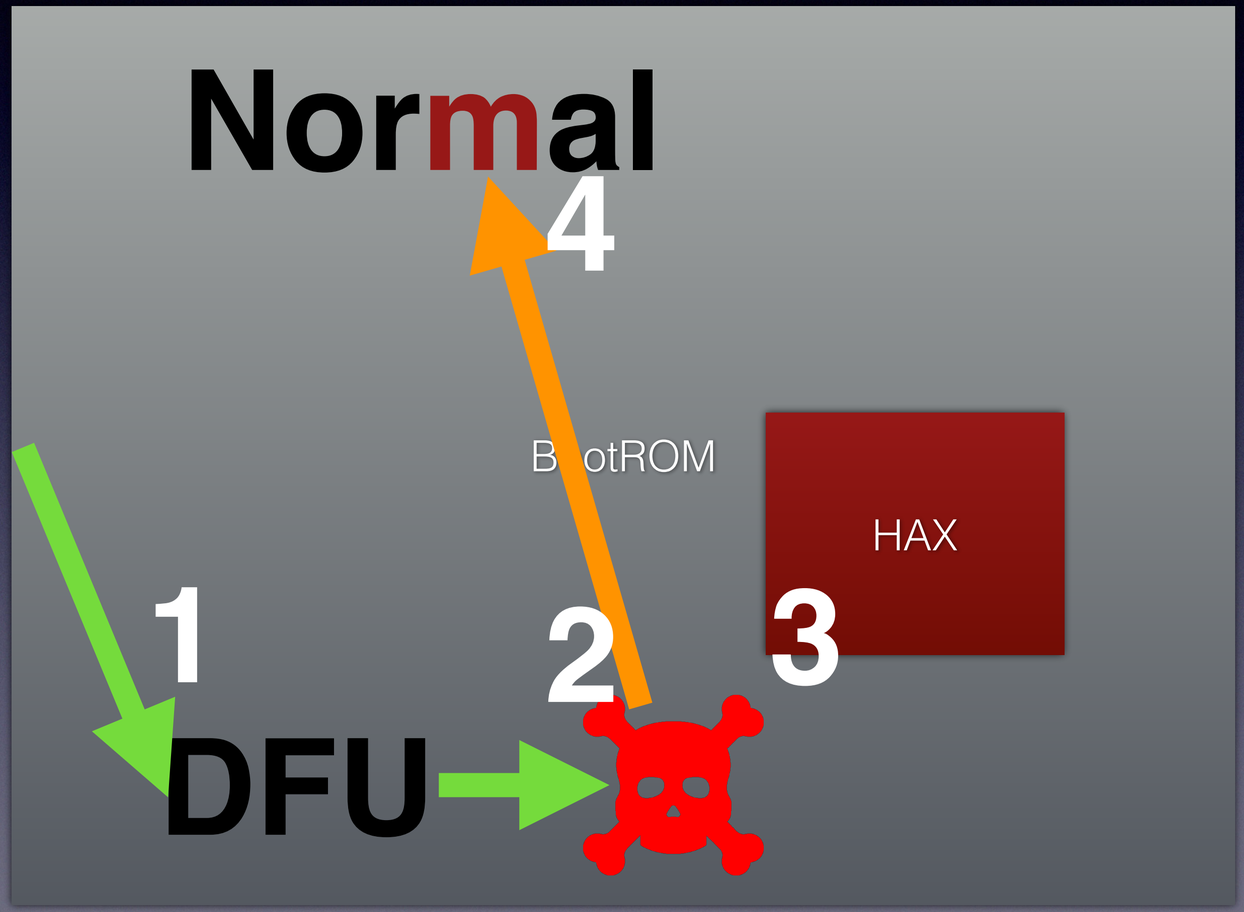
checkra1n in a Nutshell

- Enter DFU mode
- Exploit checkm8
  & inject Shellcode
- Remap BootROM page to SRAM
- Patch remapped copy



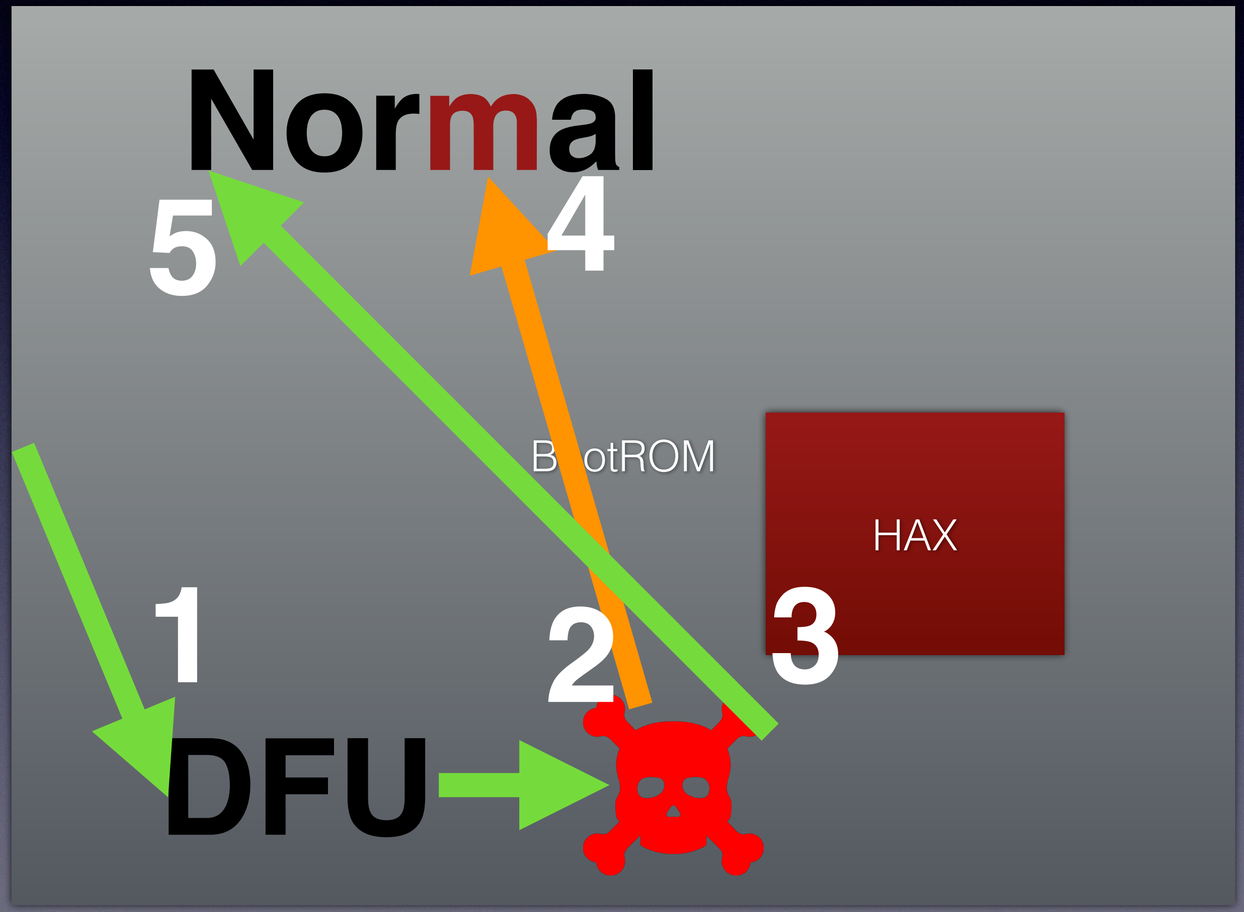
checkra1n in a Nutshell

- Enter DFU mode
- Exploit checkm8
  & inject Shellcode
- Remap BootROM page to SRAM
- Patch remapped copy



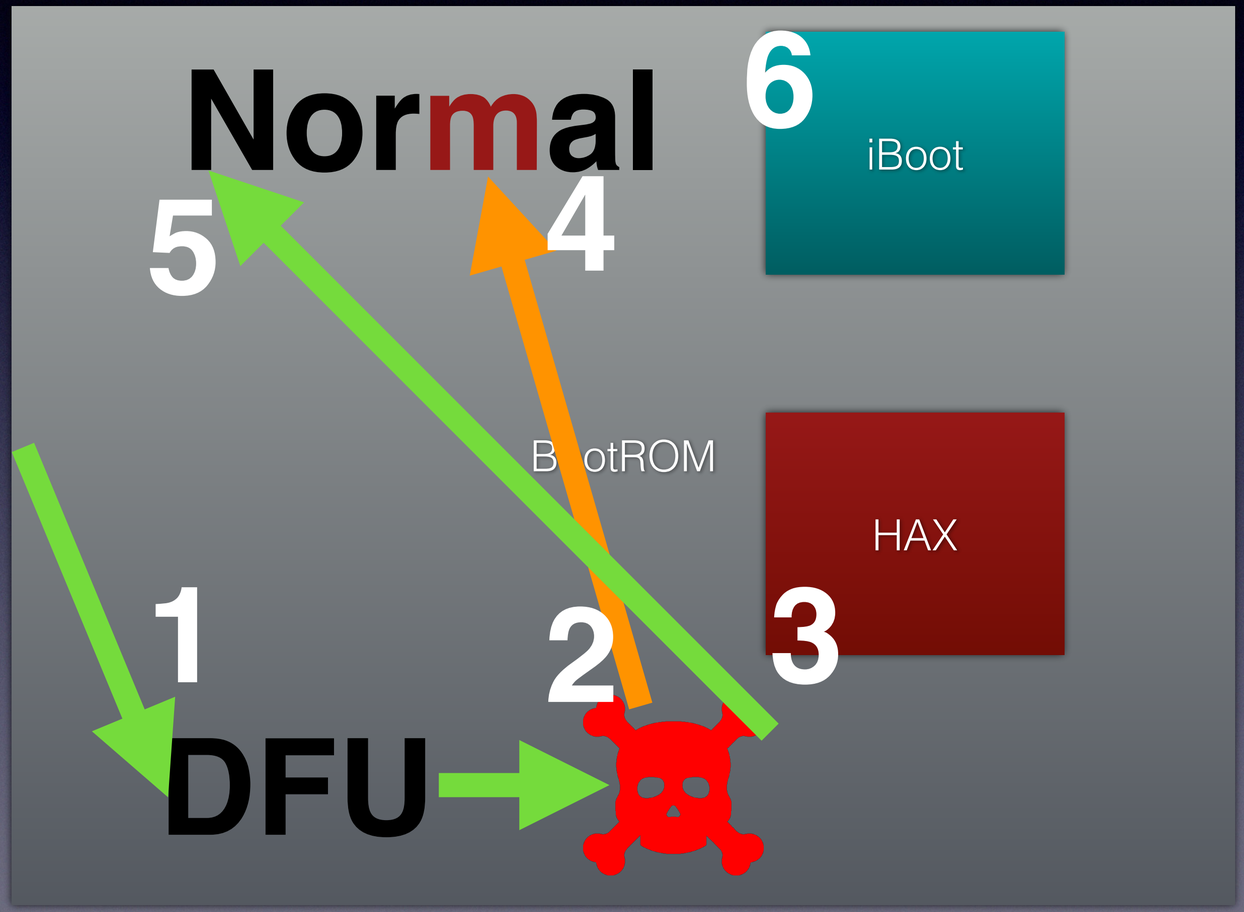
checkra1n in a Nutshell

- Jump back to normal boot
- Load iBoot from NAND & Verify it
- Patched BootROM page redirects flow to shellcode
- Shellcode patches iBoot
- Patched iBoot gets executed



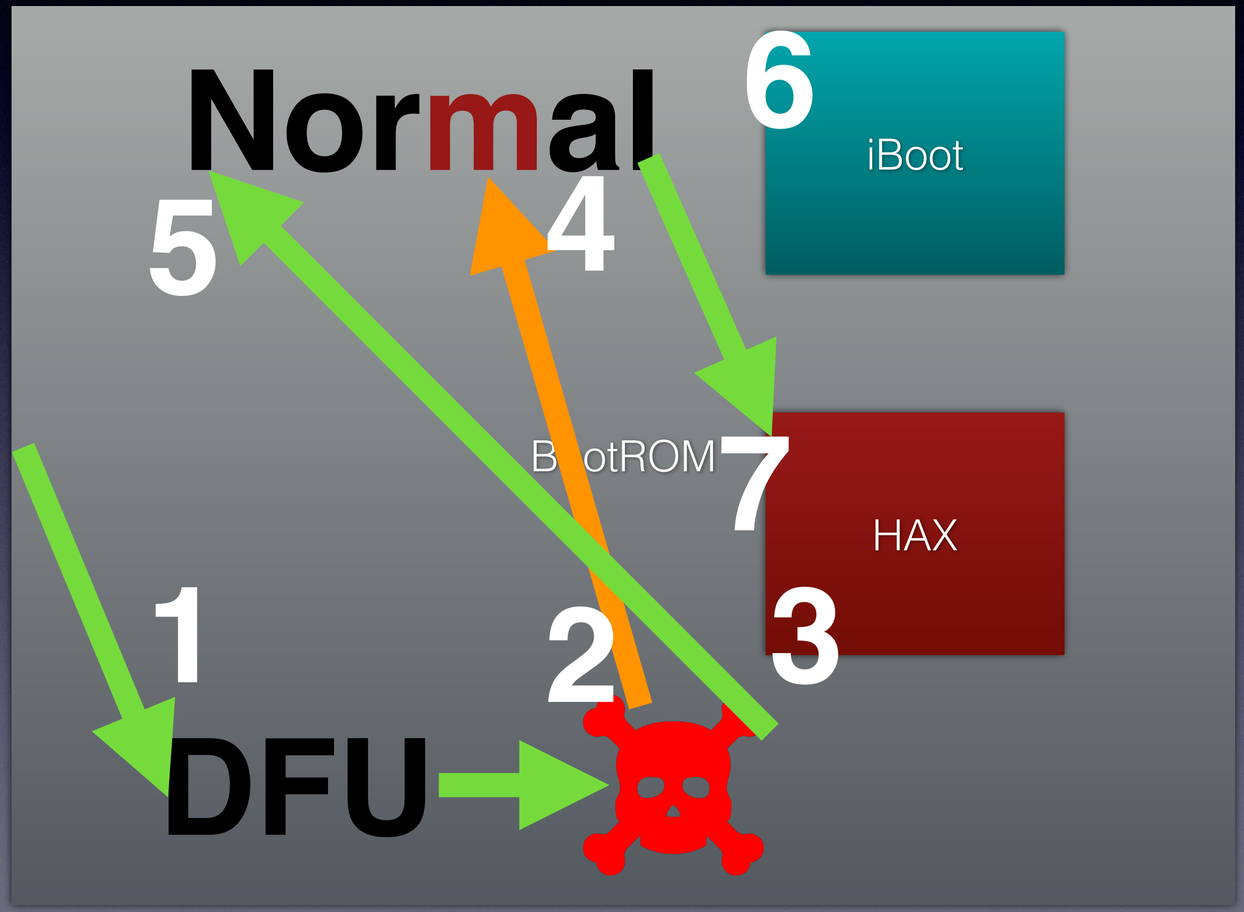
checkra1n in a Nutshell

- Jump back to normal boot
- Load iBoot from NAND & Verify it
- Patched BootROM page redirects flow to shellcode
- Shellcode patches iBoot
- Patched iBoot gets executed



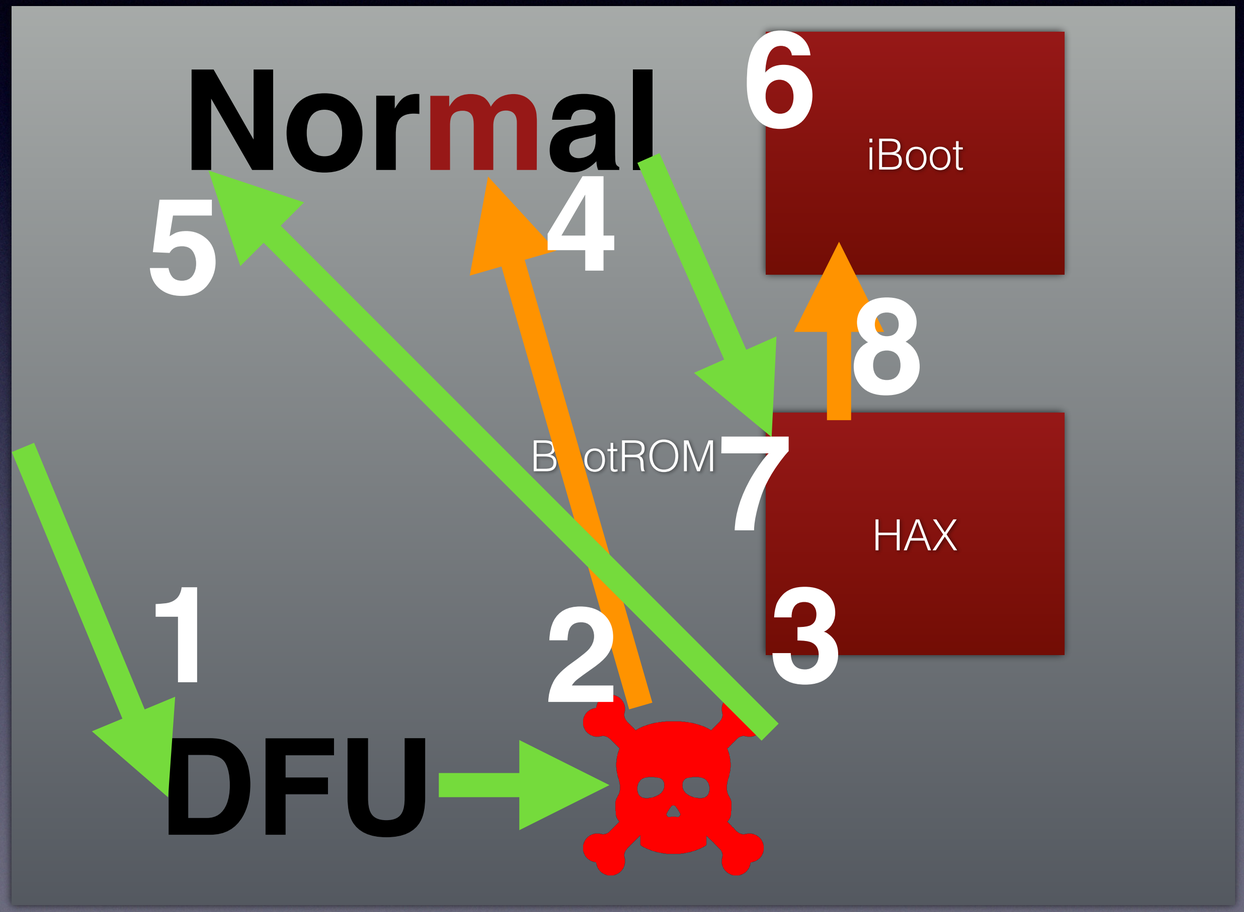
checkra1n in a Nutshell

- Jump back to normal boot
- Load iBoot from NAND & Verify it
- Patched BootROM page redirects flow to shellcode
- Shellcode patches iBoot
- Patched iBoot gets executed



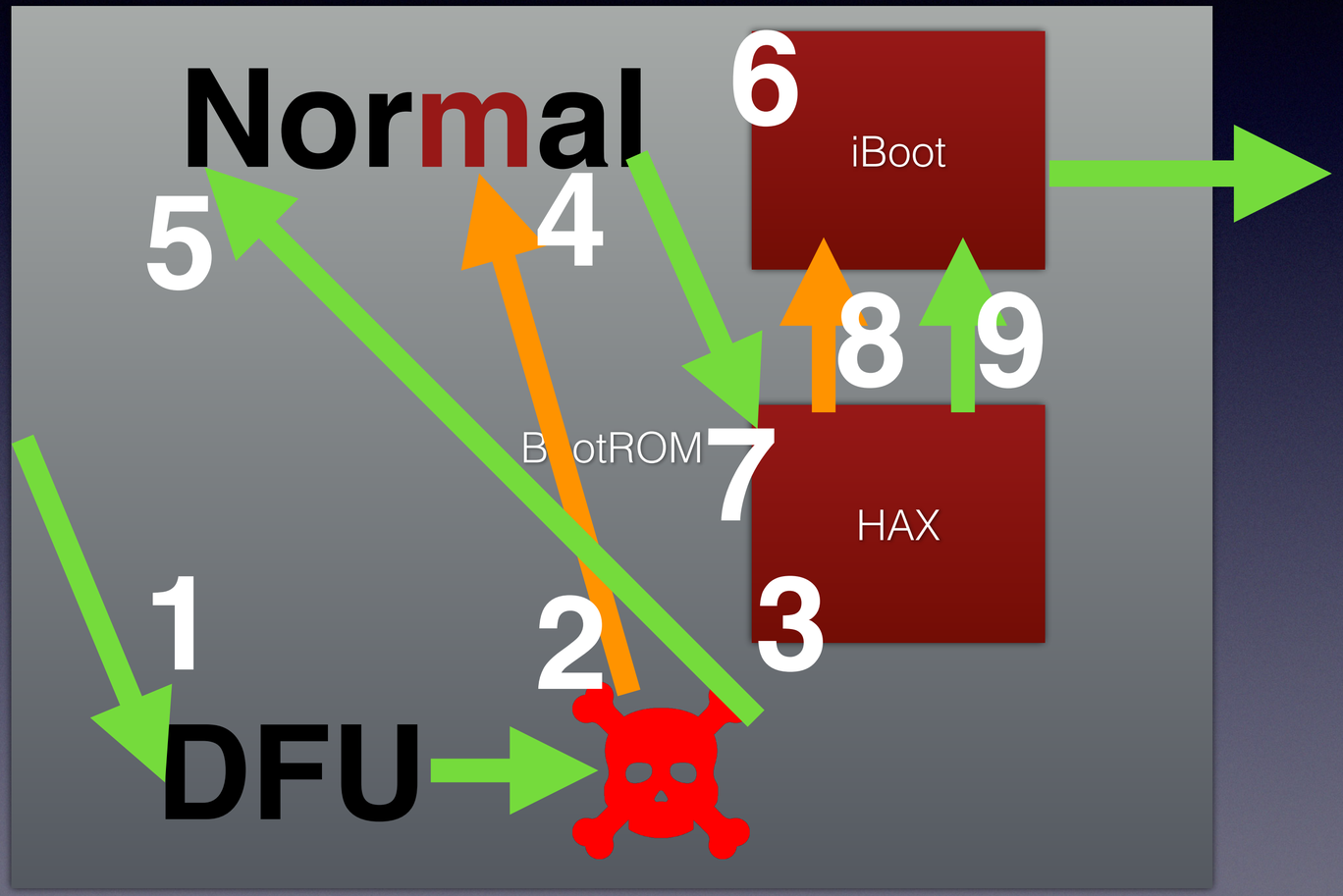
checkra1n in a Nutshell

- Jump back to normal boot
- Load iBoot from NAND & Verify it
- Patched BootROM page redirects flow to shellcode
- Shellcode patches iBoot
- Patched iBoot gets executed



checkra1n in a Nutshell

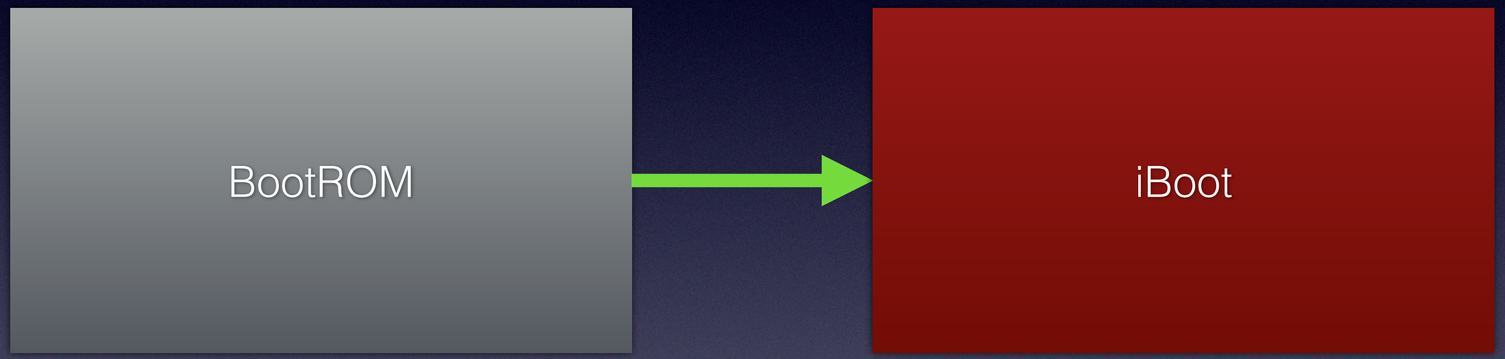
- Jump back to normal boot
- Load iBoot from NAND & Verify it
- Patched BootROM page redirects flow to shellcode
- Shellcode patches iBoot
- Patched iBoot gets executed



Notes on checkra1n

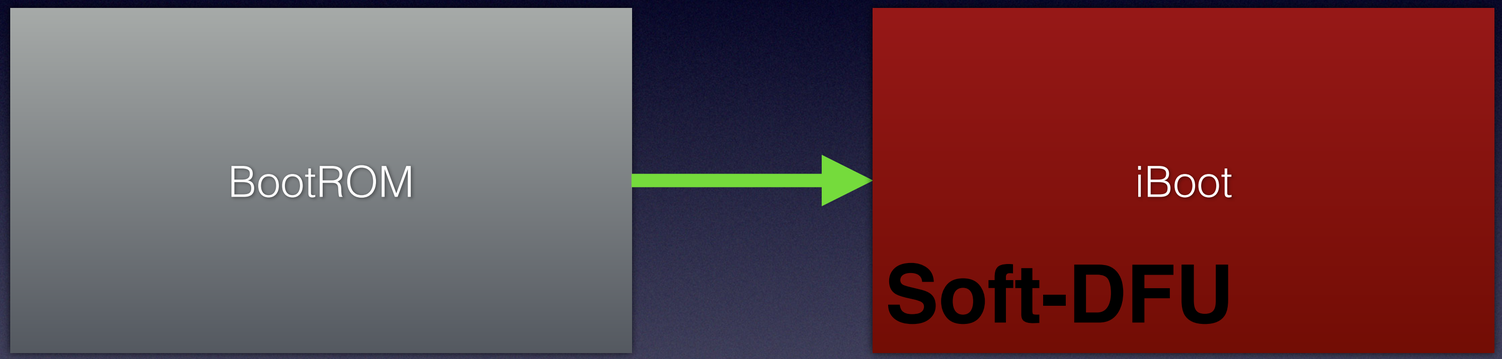
- Checkra1n never patches code signature checks
  - Patches only jumping to shellcode before booting iBoot
- Shellcode patches new iBoot
  - Patches are applied **after** code signature verification
- Checkra1n cannot boot without installed valid iBoot
  - I.e. breaks on corrupt or non-existing iBoot

checkra1n boot



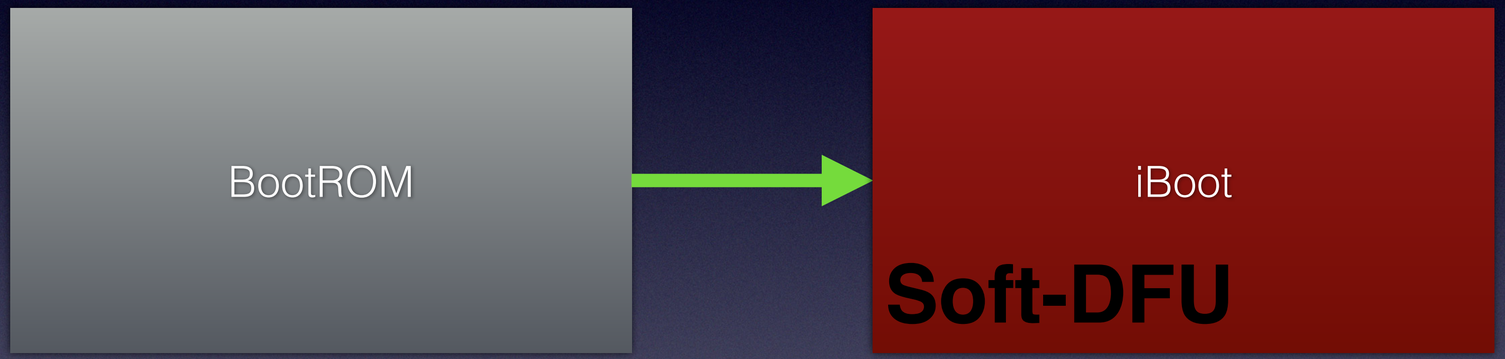
- iBoot falls back into Soft-DFU mode
- Bigger payload (pongoOS) gets uploaded (DRAM now available)
- Same procedure as in BootROM is performed for injection

checkra1n boot



- iBoot falls back into Soft-DFU mode
- Bigger payload (pongoOS) gets uploaded (DRAM now available)
- Same procedure as in BootROM is performed for injection

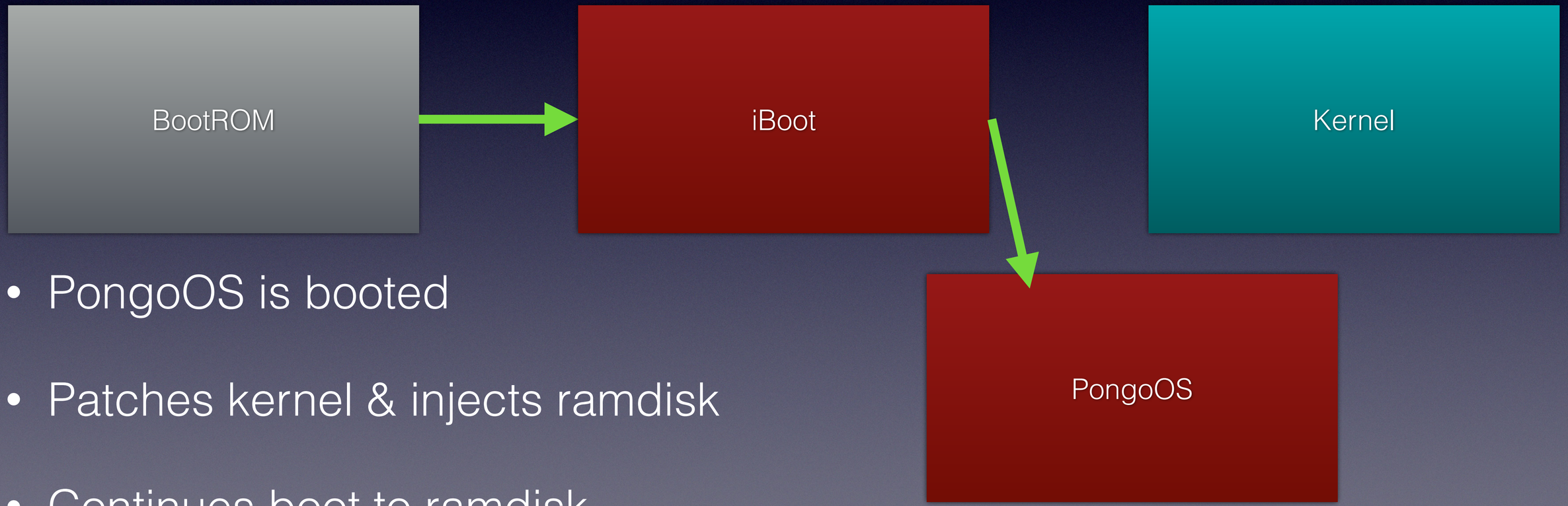
checkra1n boot



- iBoot falls back into Soft-DFU mode
- Bigger payload (pongoOS) gets uploaded (DRAM now available)
- Same procedure as in BootROM is performed for injection

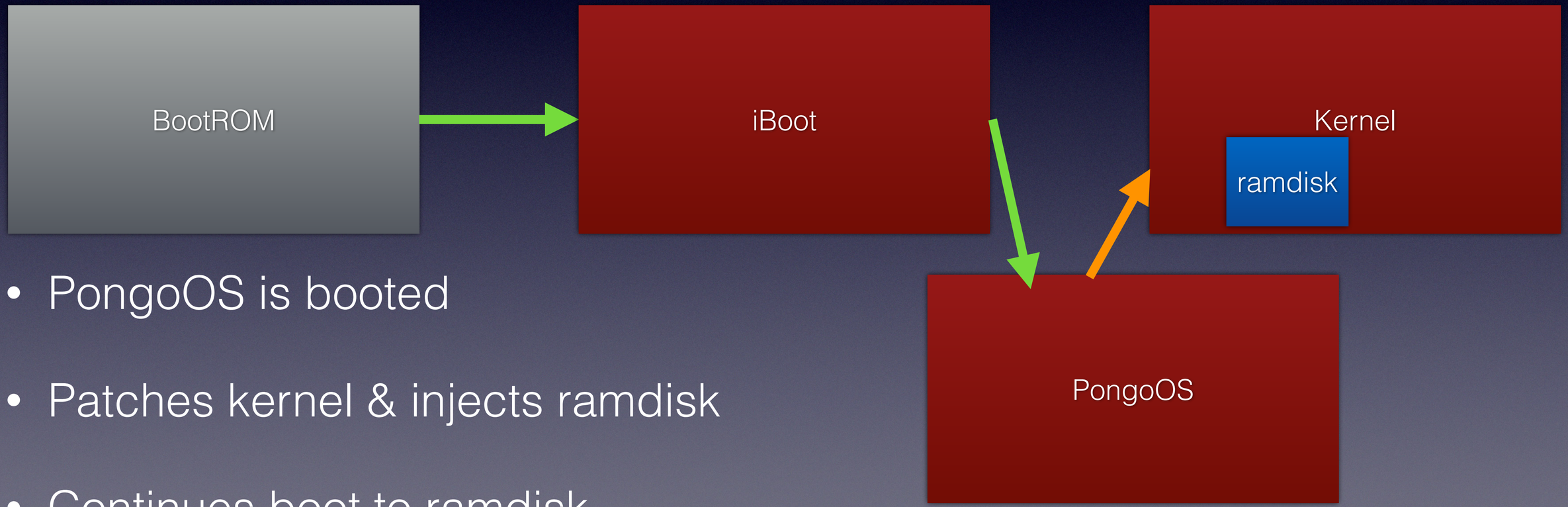


checkra1n boot



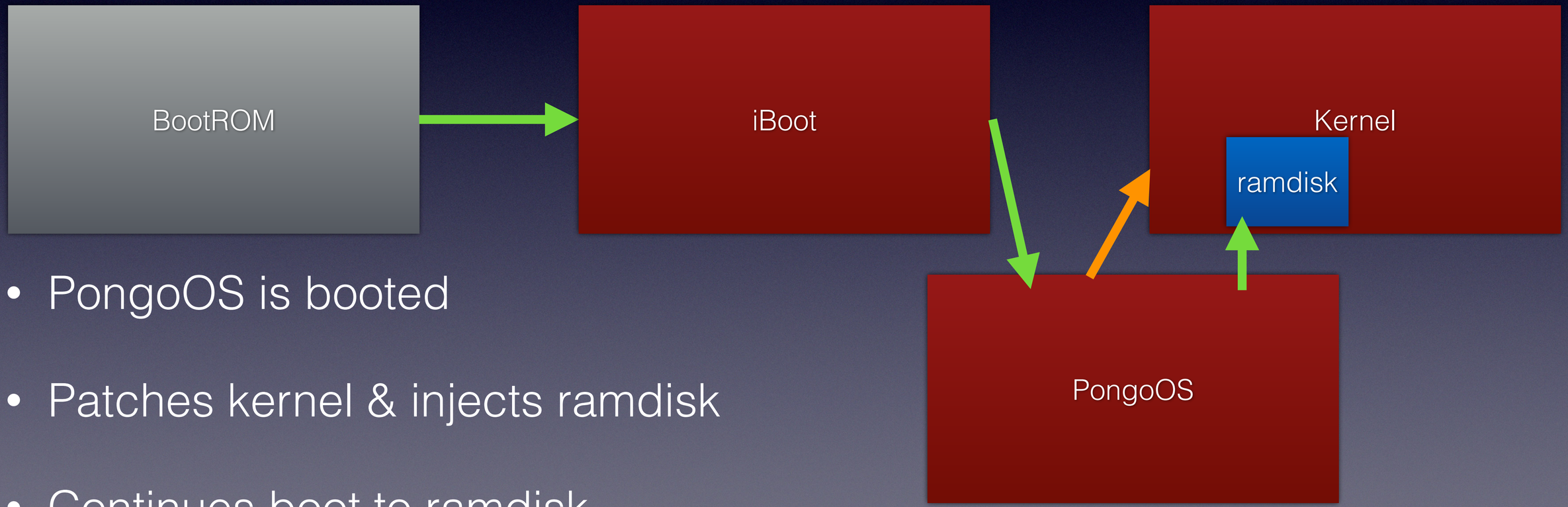
- PongoOS is booted
- Patches kernel & injects ramdisk
- Continues boot to ramdisk

checkra1n boot



- PongoOS is booted
- Patches kernel & injects ramdisk
- Continues boot to ramdisk

checkra1n boot



- PongoOS is booted
- Patches kernel & injects ramdisk
- Continues boot to ramdisk

Checkra1n Ramdisk

- Mounts real rootfs on top of ramdisk with UNION mount
- Replaces real existing daemon (by mounting over /usr/libexec)
- Continue system boot, wait for userdata unlock, upload bootstrap, enable ssh

SEP

- iOS 14 SEP adds mitigation to block jailbreak
  - SEP detects DFU boot and disables access to FS encryption keys
- PongoOS exploits A10 SEPROM and patches SEPOS to skip check
- A11 workaround:
  - iOS 14/15 disable passcode
  - iOS 16 never set passcode after erase restore
- Not applicable to <=A9
- Not relevant for HomePod (A8)

Checkra1n

- Problems with iOS 15
  - **S**igned **S**ystem **V**olume prevents rootfs modification (political problem: rootless bootstrap?)
  - Union mounts removed from kernel (technical problem)
  - iBoot removes soft-DFU (technical problem)

Checkra1n

- Problems with iOS 15
  - **S**igned **S**ystem **V**olume prevents
    (political problem: rootless boots
  - Union mounts removed from kerr
    (technical problem)
  - iBoot removes soft-DFU
    (technical problem)

HomePod feature, not used in iPhone

still exists in tvOS 16

HomePod DFU

- Connect USB
- Flip HomePod upside-down
- Power on

HomePod DFU

- BootROM boots
- iBoot boots
  - Detects that HomePod is upside down + USB connected
  - Reboots into BootROM DFU mode
- Sending iBoot over USB to BootROM enters iBoot Soft-DFU mode (if still upside down + USB connected)
  - Patch iBoot to avoid this
  - Flip HomePod to the side before sending iBoot

Ra1nsn0w

Ra1nsn0w boot

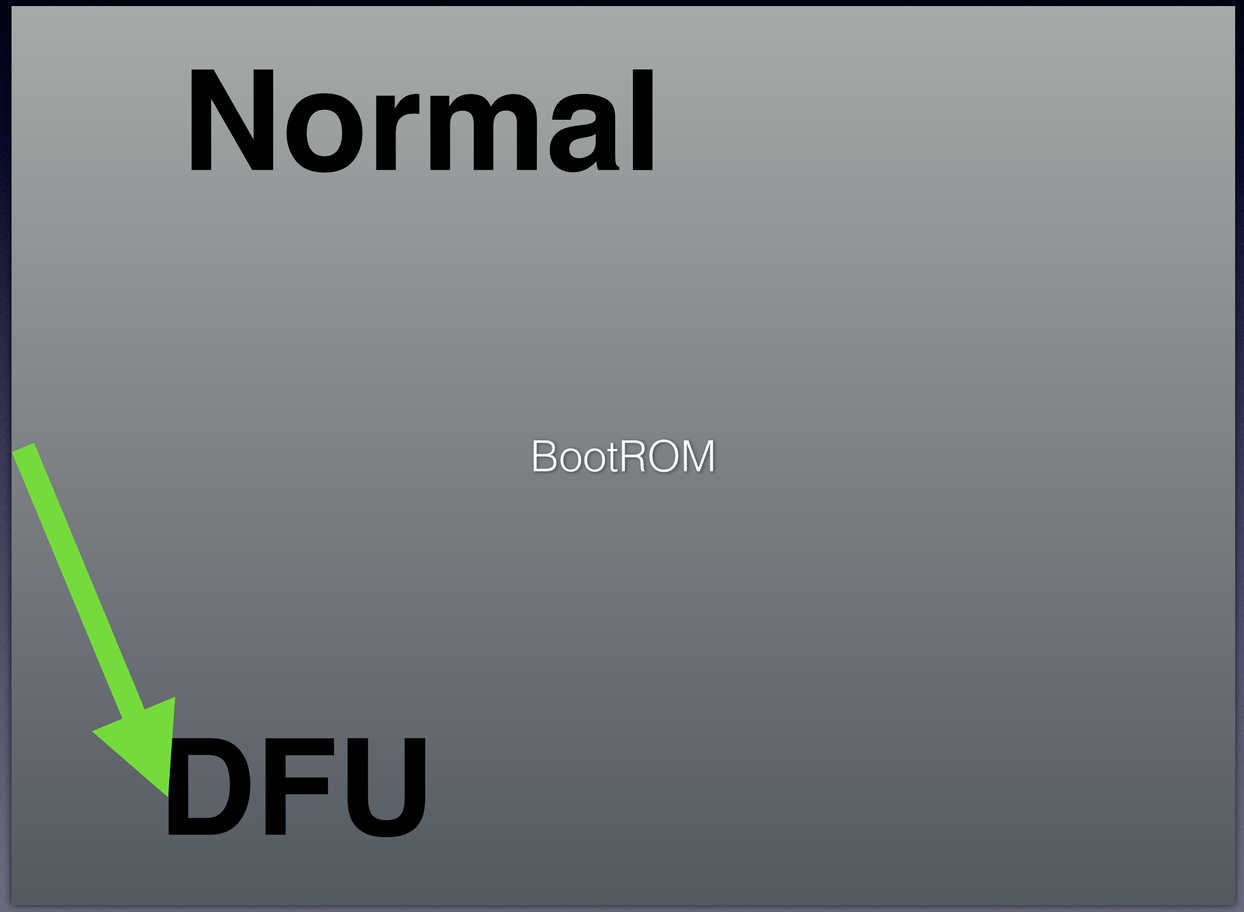
- Relies on "classic" pwnDFU mode
  - Signature checks disabled in BootROM
  - Normal (unsigned) image can be booted
- Performs limera1n-style tethered boot
  - Sending already patched components over USB (restore mode boot)
- Doesn't rely on anything from NAND
- Can boot any version iBoot/Ramdisk/Kernel regardless of what is installed

Disabling BootROM sigpatches

- Fix ipwndfu exploit to not corrupt heap
  (eg. ipwndfu_public by @LinusHenze)
- Add 2nd level page table
- Remap 1 page in BootROM to SRAM
- Apply sigpatches to remapped page
- Approach Needs:
  1 unused page (remap) + 1 partially unused page (page table)

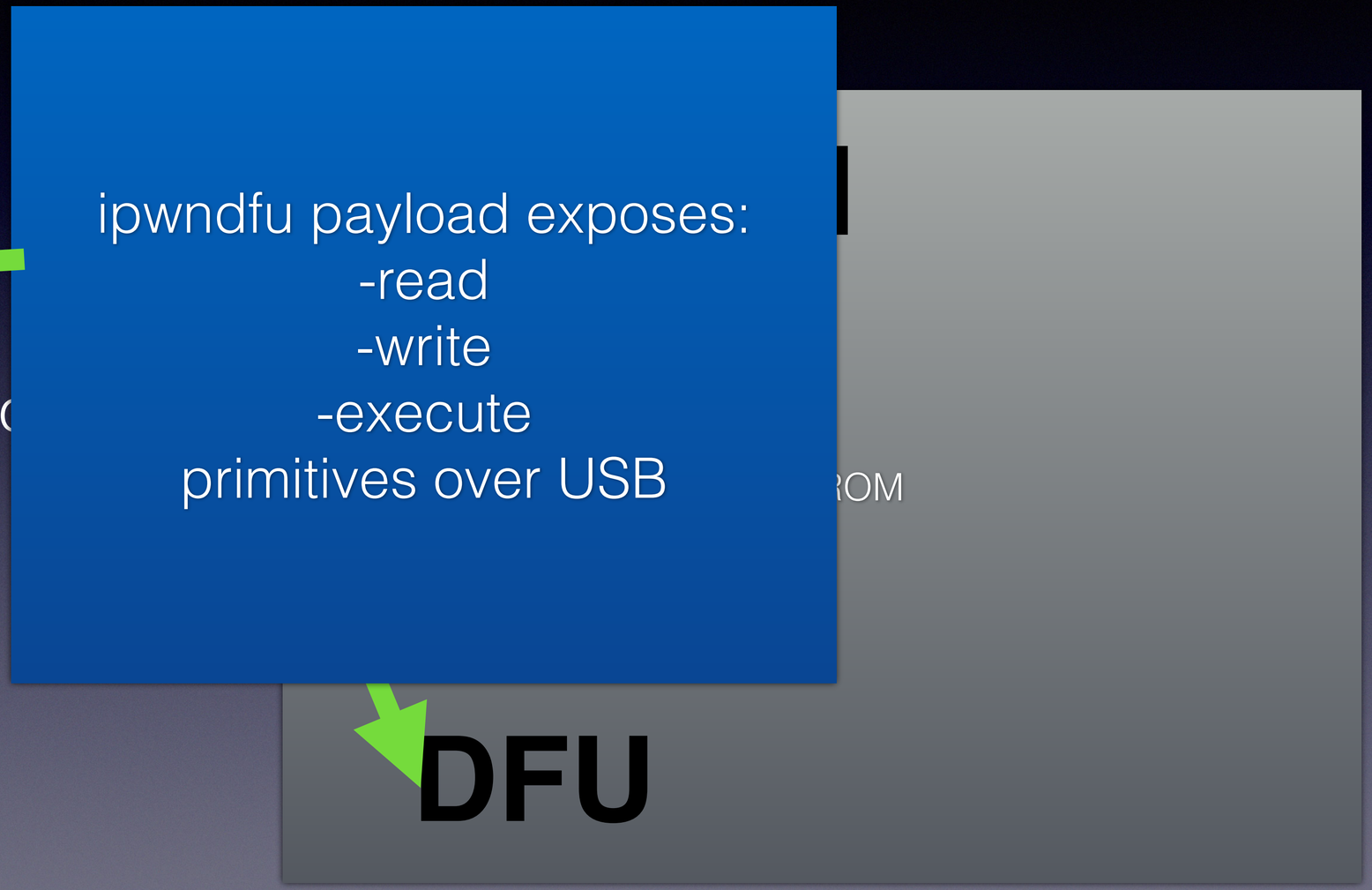
BootROM sigpatches

- Enter DFU mode
- Exploit checkm8
- Remap BootROM page to SRAM
- Patch remapped copy
- Send iBoot over USB
- Boot iBoot



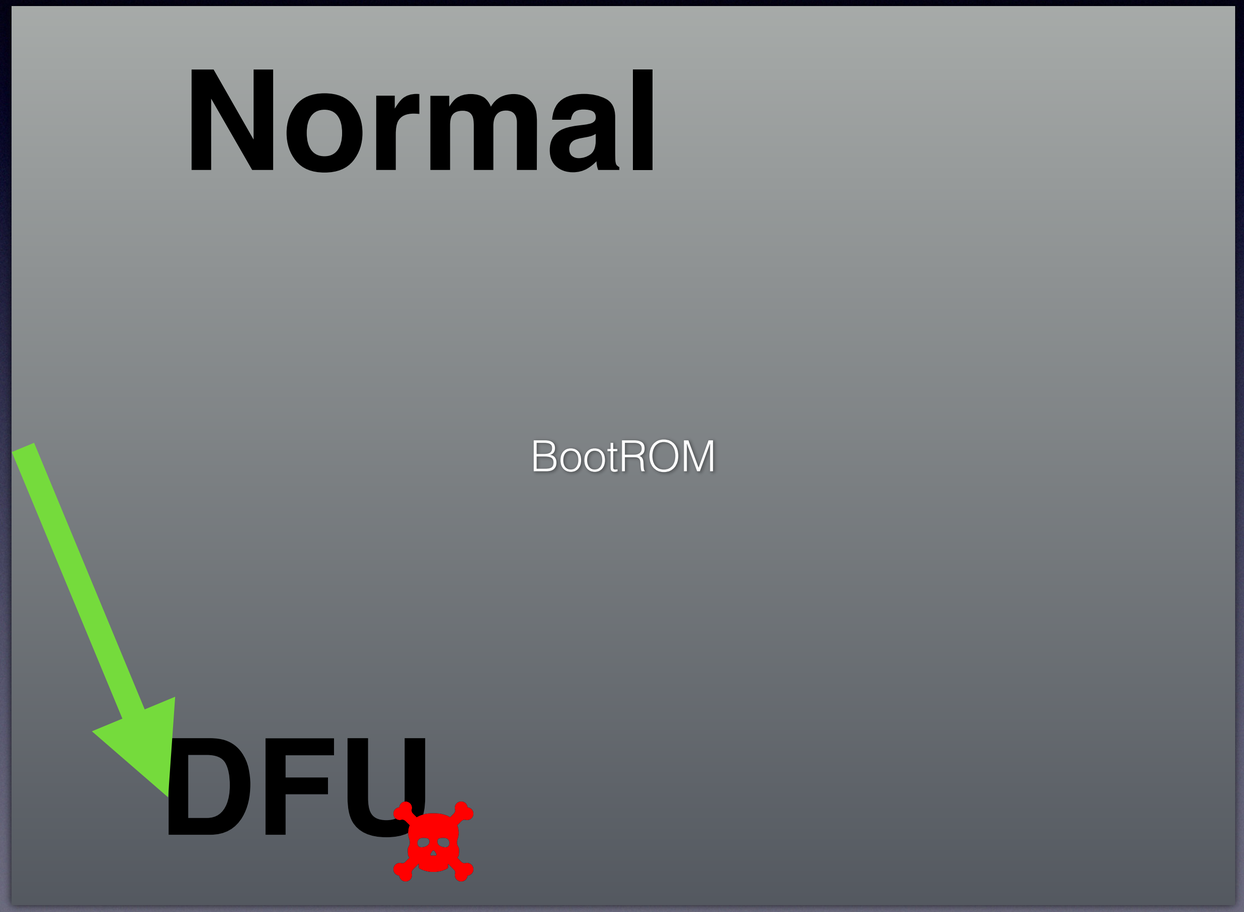
BootROM sigpatches

- Enter DFU mode
- Exploit checkm8
- Remap BootROM page to
- Patch remapped copy
- Send iBoot over USB
- Boot iBoot



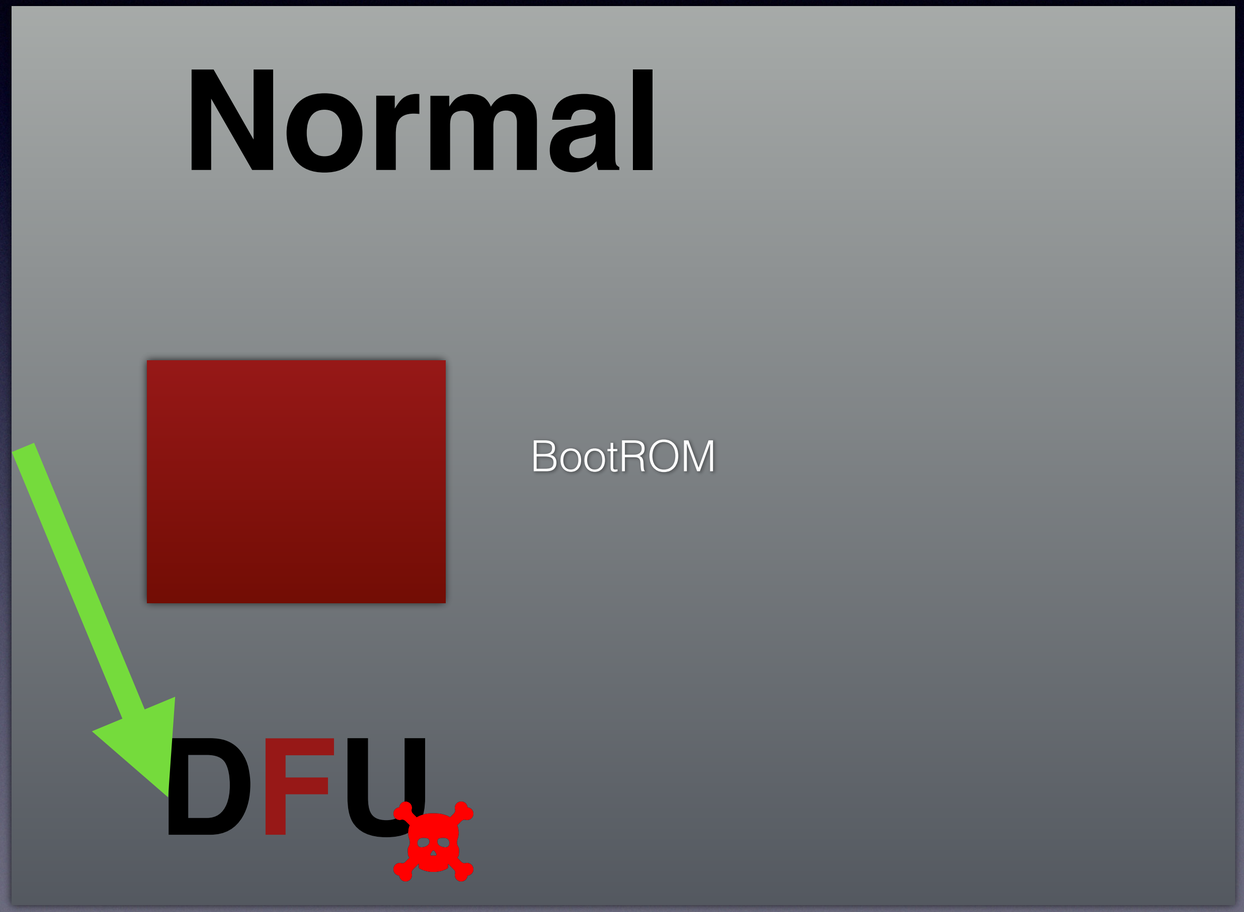
BootROM sigpatches

- Enter DFU mode
- Exploit checkm8
- Remap BootROM page to SRAM
- Patch remapped copy
- Send iBoot over USB
- Boot iBoot



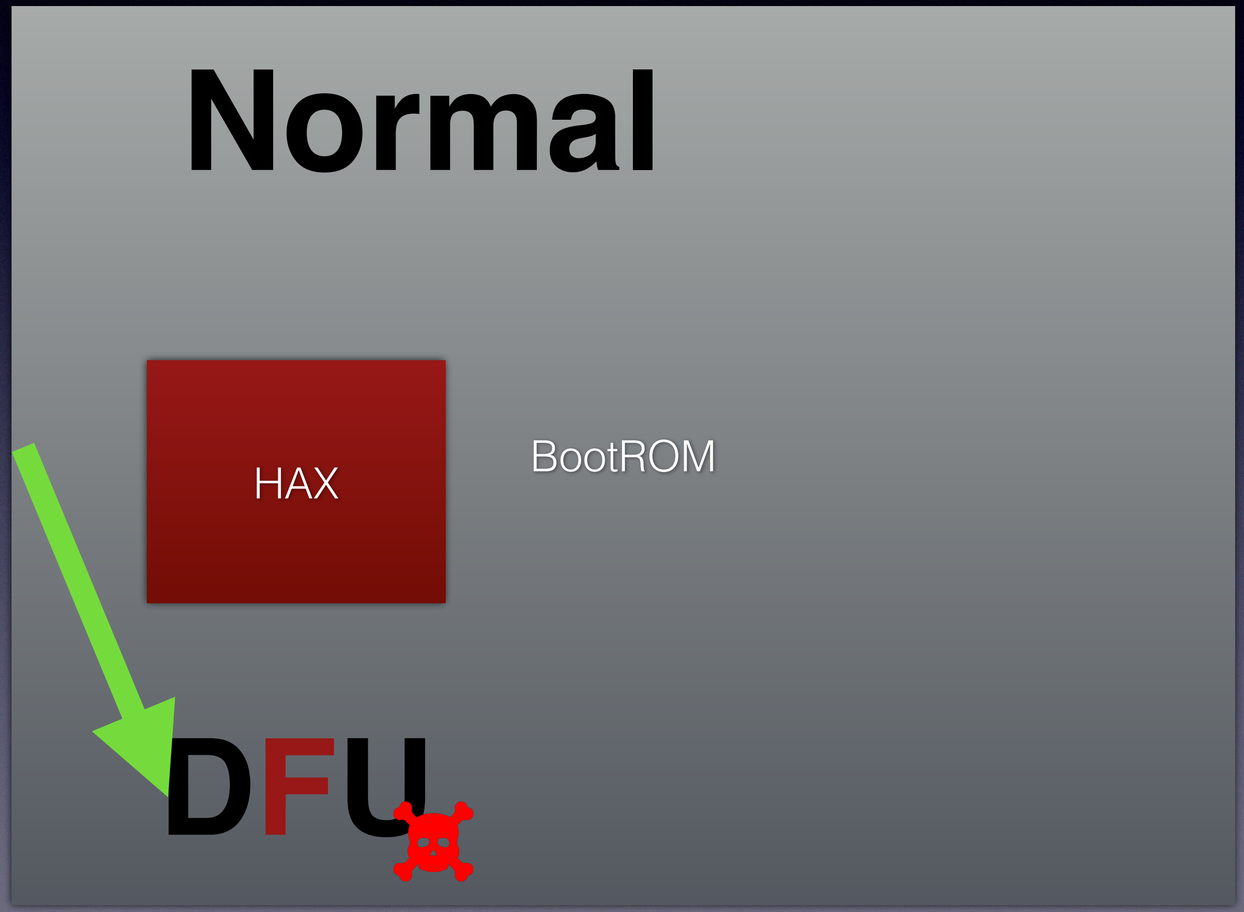
BootROM sigpatches

- Enter DFU mode
- Exploit checkm8
- Remap BootROM page to SRAM
- Patch remapped copy
- Send iBoot over USB
- Boot iBoot



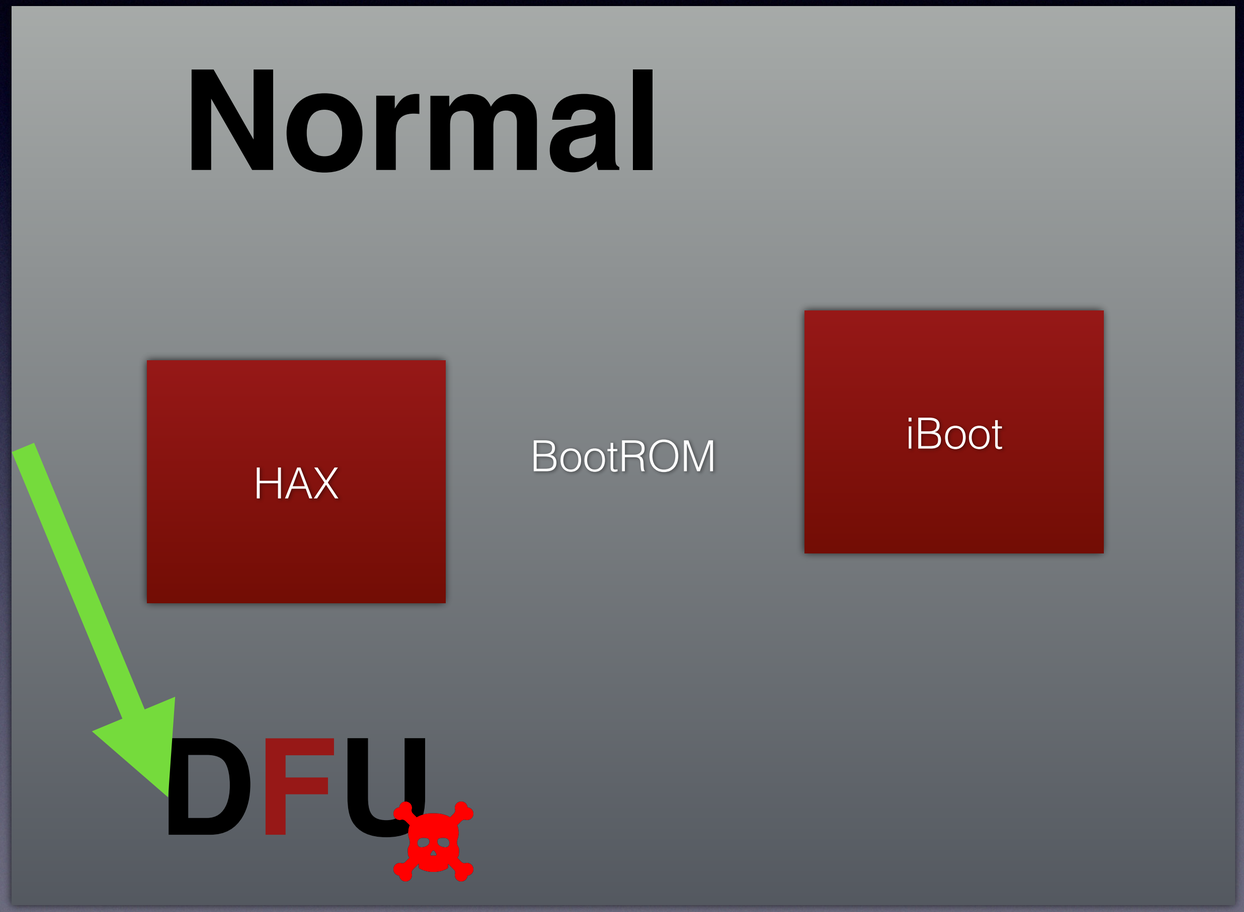
BootROM sigpatches

- Enter DFU mode
- Exploit checkm8
- Remap BootROM page to SRAM
- Patch remapped copy
- Send iBoot over USB
- Boot iBoot



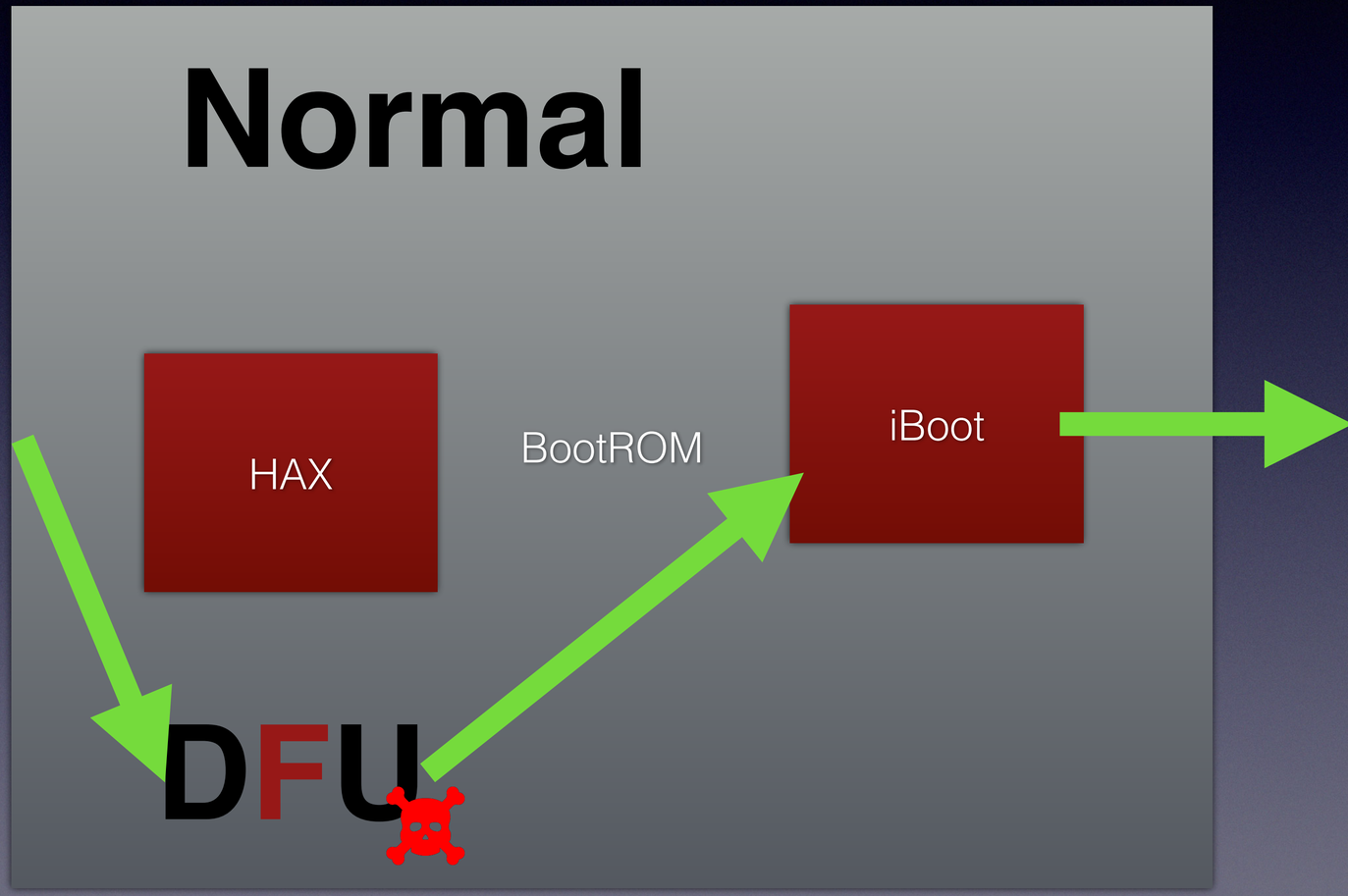
BootROM sigpatches

- Enter DFU mode
- Exploit checkm8
- Remap BootROM page to SRAM
- Patch remapped copy
- Send iBoot over USB
- Boot iBoot



BootROM sigpatches

- Enter DFU mode
- Exploit checkm8
- Remap BootROM page to SRAM
- Patch remapped copy
- Send iBoot over USB
- Boot iBoot



Ra1nsn0w boot



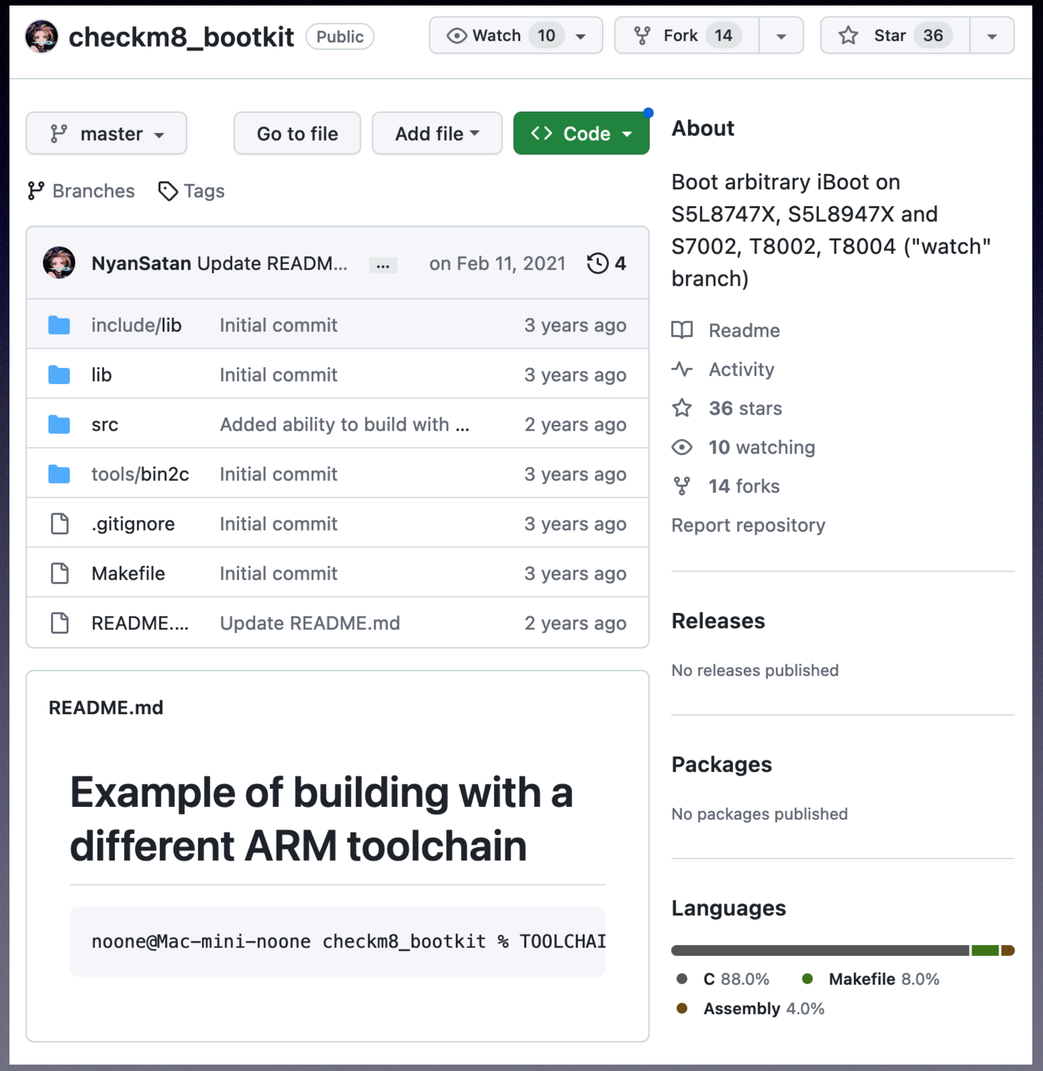
- Disable signature checks in BootROM
- Send patched iBoot
- Send patched kernel, ramdisk & all other components

Ra1nsn0w

- Component patching is performed offline (on PC)
- Patchfinder can be debugged easily for every version without needing a device
- Enable/Disable builtin patches from commandline
- Specify custom patches from commandline
  - --patch=kern:0xFFFFFFF0071CC19C,680080d208000139
- Useful for debugging/research

BootROM custom unsigned boot

- 3rd way of BootROM booting
- Don't remap BootROM, no sigpatches
- Send unsigned image over USB
- Manually call BootROM functions to prepare image (skip sigcheck)
- Redirect execution flow to image manually using exec primitive calls



Ramdisk

jbinit

- First userspace process is **/sbin/launchd**
- All binaries need to be dynamically linked against **/usr/lib/libSystem.B.dylib**
  - Fully static binaries not supported!
- Dynamic linker is really the first piece of code to run **/usr/lib/dyld**

jbinit

- Create custom ramdisk:
  - Stage 1: /usr/lib/dyld (pure C binary, asm+syscall, no deps)
  - Stage 2: /jb.dylib (shared library)
  - Stage 3: /sbin/launchd (custom daemon)
- Kernel executed launchd, but really our custom linker runs first!

jbinit stage 1

- Syscall function with inline asm
- Allocate heap memory with mmap syscall
- Read stage 2 & 3 into memory buffer
- Mount rootfs on top of ramdisk at /
  - Normal mount, not UNION mount
  - Hides access to ramdisk files

```
__attribute__((naked))
kern_return_t thread_switch(mach_port_t new_thread,
                            int option, mach_msg_timeout_t time){
  asm(
    "movn x16, #0x3c\n"
    "svc 0x80\n"
    "ret\n"
  );
}
__attribute__((naked))
uint64_t msyscall(uint64_t syscall, ...){
  asm(
    "mov x16, x0\n"
    "ldp x0, x1, [sp]\n"
    "ldp x2, x3, [sp, 0x10]\n"
    "ldp x4, x5, [sp, 0x20]\n"
    "ldp x6, x7, [sp, 0x30]\n"
    "svc 0x80\n"
    "ret\n"
  );
}
void sleep(int secs){
  thread_switch(0,2,secs*1000);
}
int sys_dup2(int from, int to){
  return msyscall(90,from,to);
}
int stat(void *path, void *ub){
  return msyscall(188,path,ub);
}
int mkdir(void *path, int mode){
  return msyscall(136,path,mode);
}
int chroot(void *path){
  return msyscall(61,path);
}
int mount(char *type, char *path, int flags, void *data){
  return msyscall(167,type,path,flags,data);
}
```

jbinit stage 1

- Write next stages to real rootfs
  - Stage 2 to /jb.dylib
  - Stage 3 to /jbinit
- Set **DYLD_INSERT_LIBRARIES** env var
- execve() into real launchd
- Stage 2 **jb.dylib** gets injected into launchd

```
printf("about to prepare execve...\n");
memset(dylib_data, 0, 0x4000);
uint8_t *bbuf = (uint8_t*)dylib_data;
char **argv = (char **)&bbuf[0];
char **envp = (char **)&bbuf[0x1000];
char *strbuf =  (char*)&bbuf[0x2000];
char *envbuf =  (char*)&bbuf[0x3000];
argv[0] = strbuf;
argv[1] = NULL;
memcpy(strbuf,"/sbin/launchd",sizeof("/sbin/launchd"));
envp[0] = envbuf;
envp[1] = NULL;
char envvars[] = "DYLD_INSERT_LIBRARIES=/jb.dylib";
memcpy(envbuf,envvars,sizeof(envvars));
printf("did prepare execve...\n");
puts("Closing console, goodbye!\n");
/*
  Launchd doesn't like it when the console is open already
*/
for (size_t i = 0; i < 10; i++) {
  close(i);
}
int err = execve(argv[0],argv,envp);
if (err) {
  printf("execve FAILED with err=%d!\n",err);
  spin();
}
```

jbinit stage 2

- LaunchDaemons are no longer loaded from plist files on filesystem
  - Cache inside dyld_shared_cache is used
  - Gets loaded as XPC dict
- Hook **xpc_dictionary_get_value,** check for **LaunchDaemons** key, inject additional daemon (Stage 3)
- Stage 3 gets loaded as launch daemon

```
xpc_object_t my_xpc_dictionary_get_value(xpc_object_t dict, const char *key){
  xpc_object_t retval = xpc_dictionary_get_value(dict,key);
  if (strcmp(key,"LaunchDaemons") == 0) {
    xpc_object_t submitJob = xpc_dictionary_create(NULL, NULL, 0);
    xpc_object_t programArguments = xpc_array_create(NULL, 0);

    xpc_array_append_value(programArguments, xpc_string_create("/jbloader"));

    xpc_dictionary_set_bool(submitJob, "KeepAlive", false);
    xpc_dictionary_set_bool(submitJob, "RunAtLoad", true);
    xpc_dictionary_set_string(submitJob, "UserName", "root");
    xpc_dictionary_set_string(submitJob, "Program", "/jbloader");
    xpc_dictionary_set_string(submitJob, "Label", "jbloader");
    xpc_dictionary_set_value(submitJob, "ProgramArguments", programArguments);

    xpc_dictionary_set_value(retval, "/System/Library/LaunchDaemons/net.tihmstar.
  }
  return retval;
}
DYLD_INTERPOSE(my_xpc_dictionary_get_value, xpc_dictionary_get_value);
```

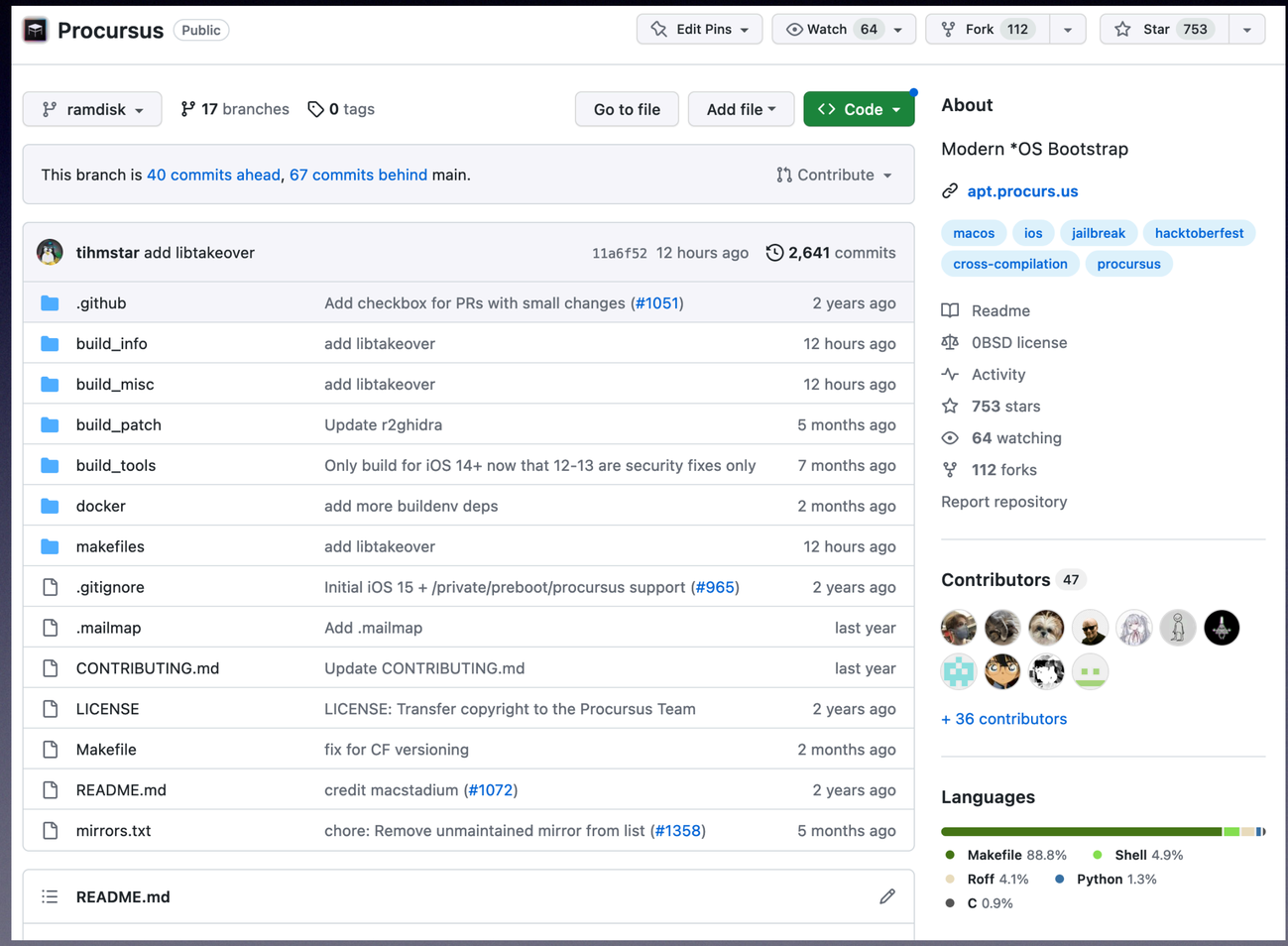
jbinit stage 3

- Open TCP socket
- Receive statically linked **tar** binary
- Receive **bootstrap.tar**
- Extract bootstrap to /
- Start SSH

```
cretassure((connfd = accept(serverfd, (struct sockaddr*)&client, (sockl
info("[deployFiles] Accepted client connection for data!");
{
  int fd_bin = -1;
  cretassure((fd_bin = open(data, O_CREAT | O_WRONLY | O_TRUNC, 0755))
  char buf[0x400];
  size_t len = 0;
  size_t didRead = 0;
  while ((len = read(connfd, buf, sizeof(buf))) > 0) {
    didRead += len;
      write(fd_bin, buf, len);
  }
  info("wrote %d bytes data\n",didRead);
  close(fd_bin);
}
const char *args[] = {
  binary,
  "--preserve-permissions",
  "-xkvf",
  data,
  "-C",
  "/",
  NULL
};
run(args[0],args);
loadDaemons();
{
  char *args[]= {
                "/bin/sh",
                "/usr/libexec/sshd-keygen-wrapper",
                "-p",
                "2223",
                NULL
            };
  run(args[0],args);
}
info("deployFiles ok!");
```

Compiling bootstrap

- Open source "easy to use" buildsystem for compiling various projects for iOS (& friends)
- **ramdisk**-branch has a docker container
  - Pass *OS + macOS SDK and compile bootstrap



Signed **S**ystem **V**olume

- iOS mounts snapshot on boot, not real rootfs
- Mounting real rootfs writeable (on it's own) does **not** destroy APFS snapshot
  - Previous JBs renamed snapshot so iOS can't find it
- Normal boot still works fine
  - But snapshot is automatically restored on next boot
  - All rootfs changes are lost!

File Persistance?

- Not breaking APFS seal serves as anti-brick-protection
  - HomePod does not have IPSWs, can't restore if sth goes wrong
- Evil-maid attack malware is gone on next reboot (stealth)
  - Can't run on reboot anyways, why keep around then?
- Prevent accidental wipe on reboot with "auto-boot = false" in iBoot
- Manually rename snapshot if you really need it (breaks non-jb boot)

Palera1n

- Uses PongoOS for kernelpatches
- Uses (modified) jbinit approach for ramdisk

jbinit Public

Watch 6

main 17 branches 0 tags Go to file Add file Code

staturnzz fix: rootless preferences folder creation, openssh install on rootful dd15fc7 2 weeks ago 237 commits

.github/workflows	fix: create loader.dmg from ipa	last month
dropbear-plist	Integrate cfprefsd hook	2 months ago
src	fix: rootless preferences folder creation, openssh install on rootful	2 weeks ago
.gitignore	fix: create loader.dmg from ipa	last month
.gitmodules	Update .gitmodules	last month
LICENSE	finally add license	3 months ago
Makefile	fix: create loader.dmg from ipa	last month
README.md	Integrate cfprefsd hook	2 months ago

README.md

jbinit

used in palera1n-c.

Files not included, but needed

`palera1n.ipa`: palera1n loader `binpack.tar`: Procursus binpack

Additional notes

`ASAN` builds will exceed launchd memory limits. Do not use under non-development situations.

Tweak Injection

Tweak Injection

- Need to be able to load custom code (tweaks) into every process to modify HomePod behavior
- Inject a tweak injection library into each process which will load the tweaks for that process
  - libhooker
- How to inject the tweak injection library?

posix_spawn hook

- We already have a library injected into launchd
- launchd is the parent of (almost) all processes
- Patch posix_spawn in launchd to always add DYLD_INSERT_LIBRARIES to environment variables
  - iOS/tvOS exclusively uses posix_spawn to launch processes
- Dyld will then load jbinjector.dylib during process launch which loads TweakInject.dylib
  - jbinjector also patches posix_spawn in the new process

TweakInject

- libhooker helper library
- Loads libhooker itself, which provides various APIs
  - e.g. function hooking, ObjC method hooking
- Discovers installed tweaks and loads them for current process

What can you do with a jailbroken HomePod???

File System modifications

- Some simple things can be done by replacing stuff on the file system
  - e.g. sounds
- This does not require tweak injection
- Example: Replace the "timer done" sound

File System modifications

- Demo

spotifyd

- HomePod can only be controlled from an iOS device
  - No way to play music from Android
- Spotify has an open-source daemon (spotifyd) which can be installed on a device to play music
  - Can be controlled from Android and iOS
- Install spotifyd on HomePod to play music from Android!

spotifyd Demo

- Demo

HAL9000

- Demo

HAL9000

- Hook assistantd (responsible for Siri) and SoundBoard (responsible for UI)
- Listen for specific phrases in Siri request
- Replace reply when keyword is detected
- Write "flag" to file system so tweak injected into SoundBoard knows that it should change the UI

SiriGPT

- Siri itself is kinda limited
  - "I found this on the web"
- Idea: Hook up Siri to ChatGPT to get more useful answers
- Create a Tweak that intercepts all Siri requests, extract the text and then use the OpenAI API to get an answer from ChatGPT
- Inject a text reply containing the answer from ChatGPT

SiriGPT Demo

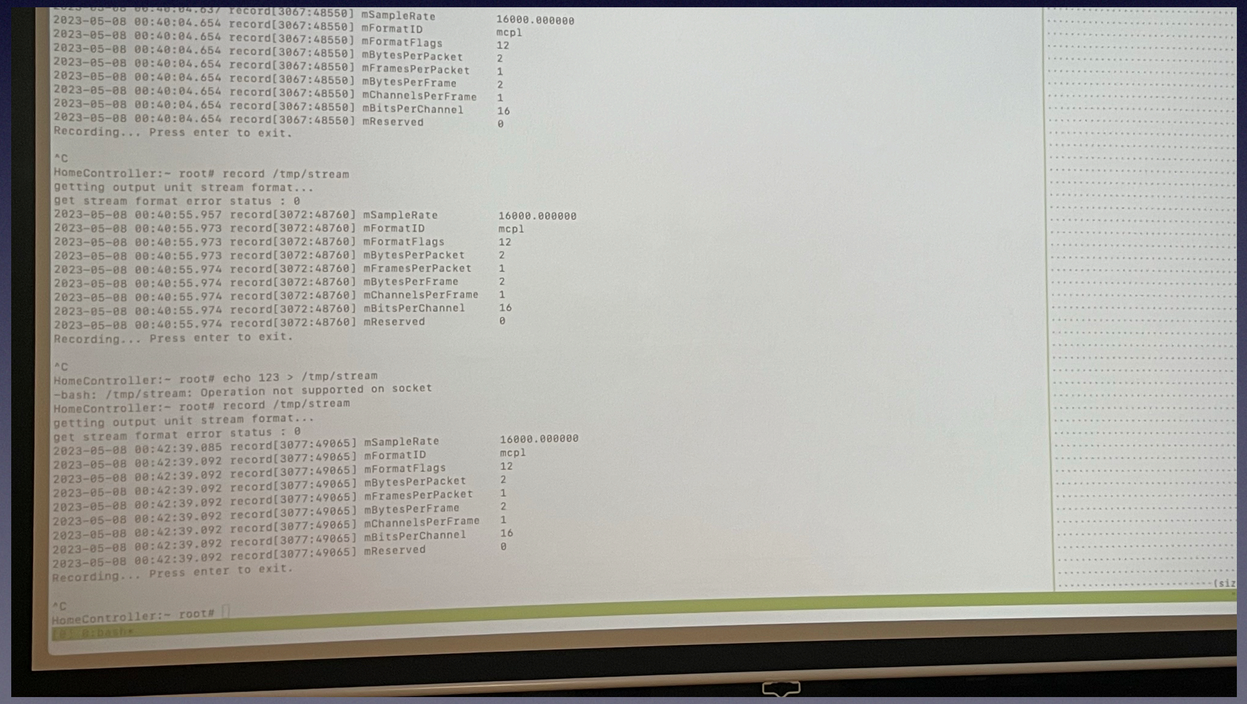
More AI in HomePod

- DEMO

micSpy

- HomePod has a microphone that we can access
  - Can be used as a bug!
- No microphone indicator
  - Spy on someone without their knowledge
- Directly stream audio data via TCP/UDP

micSpy


2023-05-08 00:40:04.654 record[3067:48550] mSampleRate 16000.000000
2023-05-08 00:40:04.654 record[3067:48550] mFormatID mcpl
2023-05-08 00:40:04.654 record[3067:48550] mFormatFlags 12
2023-05-08 00:40:04.654 record[3067:48550] mBytesPerPacket 2
2023-05-08 00:40:04.654 record[3067:48550] mFramesPerPacket 1
2023-05-08 00:40:04.654 record[3067:48550] mBytesPerFrame 2
2023-05-08 00:40:04.654 record[3067:48550] mChannelsPerFrame 1
2023-05-08 00:40:04.654 record[3067:48550] mBitsPerChannel 16
2023-05-08 00:40:04.654 record[3067:48550] mReserved 0
Recording... Press enter to exit.
^C
HomeController:~ root# record /tmp/stream
getting output unit stream format...
get stream format error status : 0
2023-05-08 00:40:55.957 record[3072:48760] mSampleRate 16000.000000
2023-05-08 00:40:55.973 record[3072:48760] mFormatID mcpl
2023-05-08 00:40:55.973 record[3072:48760] mFormatFlags 12
2023-05-08 00:40:55.973 record[3072:48760] mBytesPerPacket 2
2023-05-08 00:40:55.974 record[3072:48760] mFramesPerPacket 1
2023-05-08 00:40:55.974 record[3072:48760] mBytesPerFrame 2
2023-05-08 00:40:55.974 record[3072:48760] mChannelsPerFrame 1
2023-05-08 00:40:55.974 record[3072:48760] mBitsPerChannel 16
2023-05-08 00:40:55.974 record[3072:48760] mReserved 0
Recording... Press enter to exit.
^C
HomeController:~ root# echo 123 > /tmp/stream
-bash: /tmp/stream: Operation not supported on socket
HomeController:~ root# record /tmp/stream
getting output unit stream format...
get stream format error status : 0
2023-05-08 00:42:39.085 record[3077:49065] mSampleRate 16000.000000
2023-05-08 00:42:39.092 record[3077:49065] mFormatID mcpl
2023-05-08 00:42:39.092 record[3077:49065] mFormatFlags 12
2023-05-08 00:42:39.092 record[3077:49065] mBytesPerPacket 2
2023-05-08 00:42:39.092 record[3077:49065] mFramesPerPacket 1
2023-05-08 00:42:39.092 record[3077:49065] mBytesPerFrame 2
2023-05-08 00:42:39.092 record[3077:49065] mChannelsPerFrame 1
2023-05-08 00:42:39.092 record[3077:49065] mBitsPerChannel 16
2023-05-08 00:42:39.092 record[3077:49065] mReserved 0
Recording... Press enter to exit.
^C
HomeController:~ root#


Elias Limneos
14.132 Posts
Folge ich
Elias Limneos
@limneos Folgt Dir
iOS Developer of AnsweringMachine,BioProtect,CallBar,AudioRecorder(1st & #1 iOS call recorder) & many more. Repo: limneos.net/repo Instagram:limneos
Wissenschaft und Technik
Greece
limneos.net
Seit September 2009 bei Twitter
541 Folge ich
52.310 Follower
Gefolgt von md, opa334 und 19 weiteren Personen, denen du folgst

micSpy Demo

- Demo

Implications

- No easy way to detect a HomePod has been jailbroken
  - No persistent file system modifications
  - No way to run own code without jailbreaking it
    - And jailbreaking requires rebooting the device...

Implications

- HomePod only reboots when it is updated
  - Can stay jailbroken for a long time without anyone noticing
  - Disable updates -> HomePod will never reboot
- It takes about 5 Minutes to open up a HomePod and jailbreak it
  - Evil maid scenario
- No microphone indicator -> Spy on someone without them noticing

Implications

- EVERYBODY GET PARANOID NOW!!!!

One more thing

Less AI in HomePod

One more thing

Q&A