Husky Records
@*Northeastern University*

Cameron Kennedy
Christopher Brown
Dennis Giese
Erik Uhlmann
Trey Del Bonis

Advised By: Prof. Guevara Noubir

www.huskyrecords.net

German Engineering
Made in USA

Outline

Team and Organization

Secure System Design

RISC architecture is gonna change everything...

Attacks!

Lots of attacks

General Comments

Last year's team: DeNUvo (eCTF2019)

read more here:
www.huskyrecords.net/2019/

The Husky Records Team (eCTF 2020)

Prof. Guevara Noubir (advisor)

Dennis Giese (lead, build environment, testing, attack team)

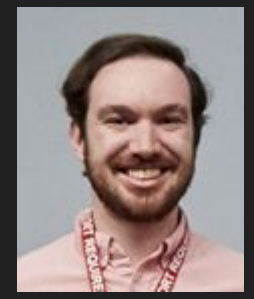
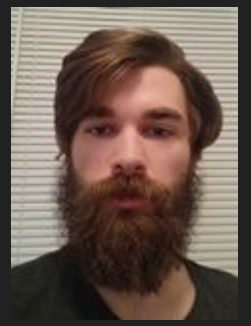
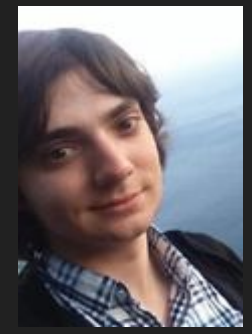
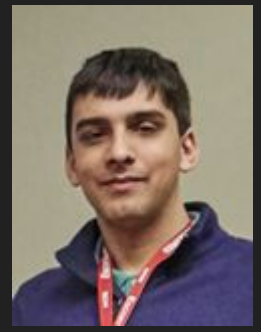
Cameron Kennedy (microblaze, crypto, attack team)

Christopher Brown (mipod, deploy scripts)

Erik Uhlmann (fpga design, crypto, attack team)

Trey Del Bonis (merkle tree, crypto, docs)



Organization

- No class, no credits
- Ordered 2 additional development boards
- In the early phase:
  - meeting every week, discussing design
  - setting up build cluster (based on 5 ESXi servers)
- 2 Days before begin of Attack phase
  - Design works, but MB is too slow
  - Crypto cannot be optimized to meet timing requirements
  - Started to redesign crypto (sleep is overrated)
- When campus was purged:
  - Distribution of boards, remote development

And I made some pancakes…

Secure System Design

Possible attack methods

- Exploit design flaws
- Exploit MB code (buffer overflow & friends)
- Hardware attacks (glitching, side channel)
- Brute force PINs or secrets
- Dump MB memory (shhh more on this later)

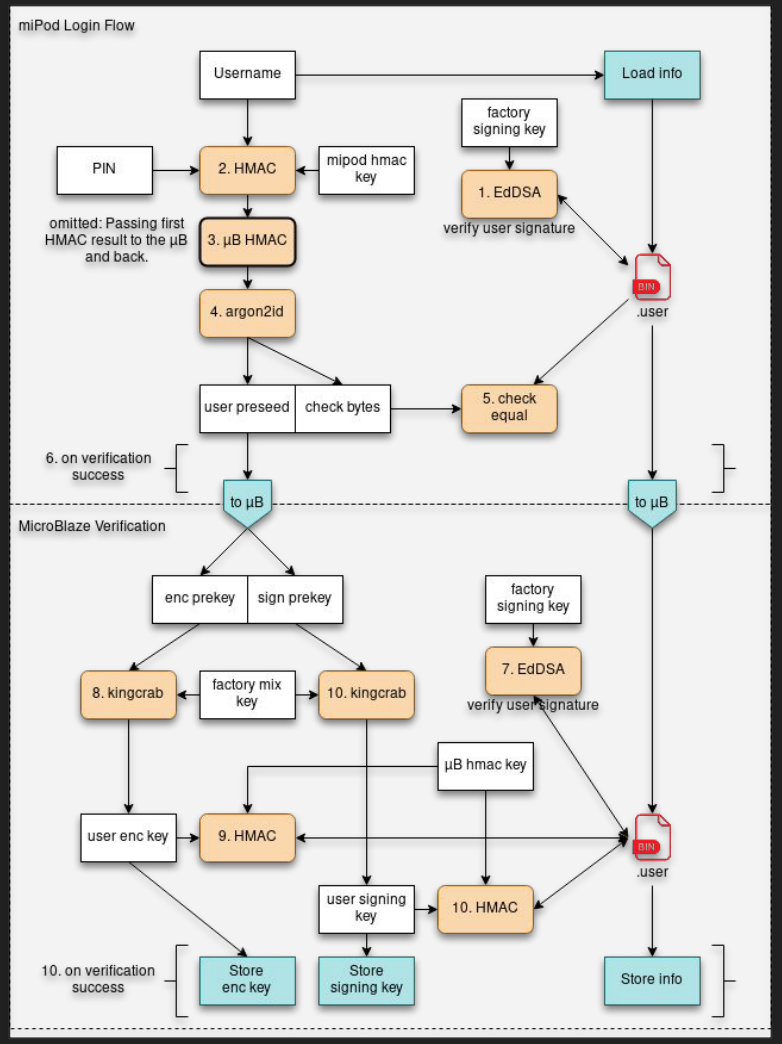
Solution: ~~Sue everyone using our EULA~~

Layer the defenses

Secure System Design

- Use modern cryptographic primitives
  - Monocypher library: HMAC-Blake2b, EdDSA, XChaCha20, Poly1305
  - Argon2id hashing
  - Blockchain technology (Blake2b Merkle trees)
- Implement capabilities with pure crypto — minimal runtime checks
- Complex key derivation processes
- Store secrets in FPGA fabric
- Protections for memory errors and glitching
- Reset on any detected anomaly
- Make reverse engineering *annoying*

Flowcharts!



Login process

Goals: authenticate users, authorize playing user songs, resistant against online/offline brute force

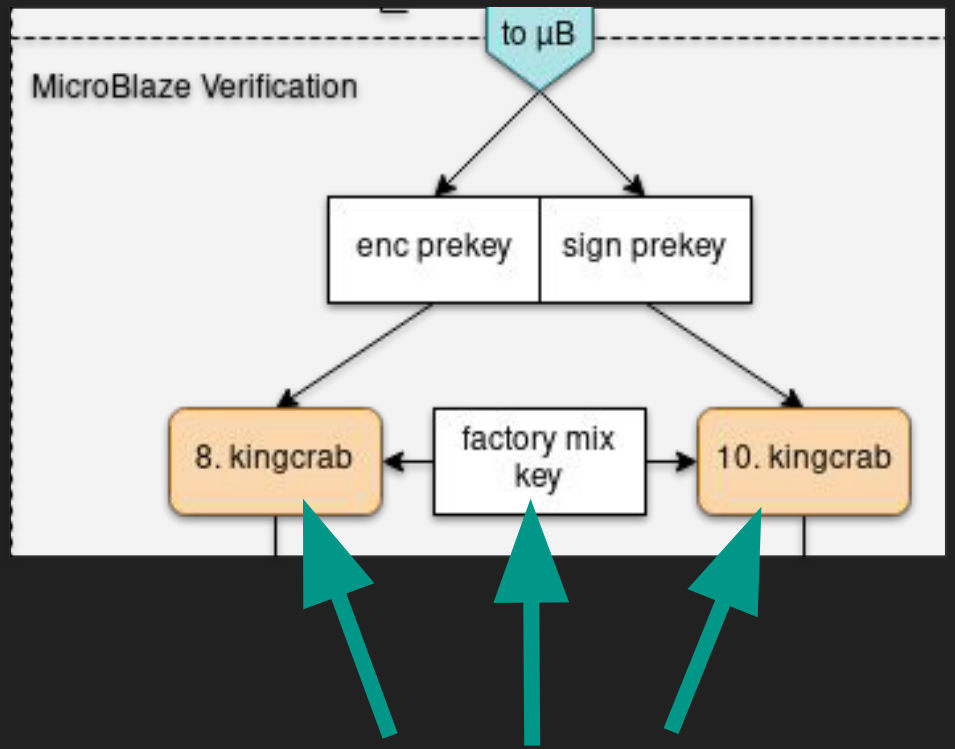
- Input: user info and entered PIN
- ARM HMAC
- MB HMAC
- Argon2id (check correct PIN)
- User info signature verification
- Mix with FPGA hardware secrets
- Derive user keys

Secret Storage

Question: where do you store secrets?

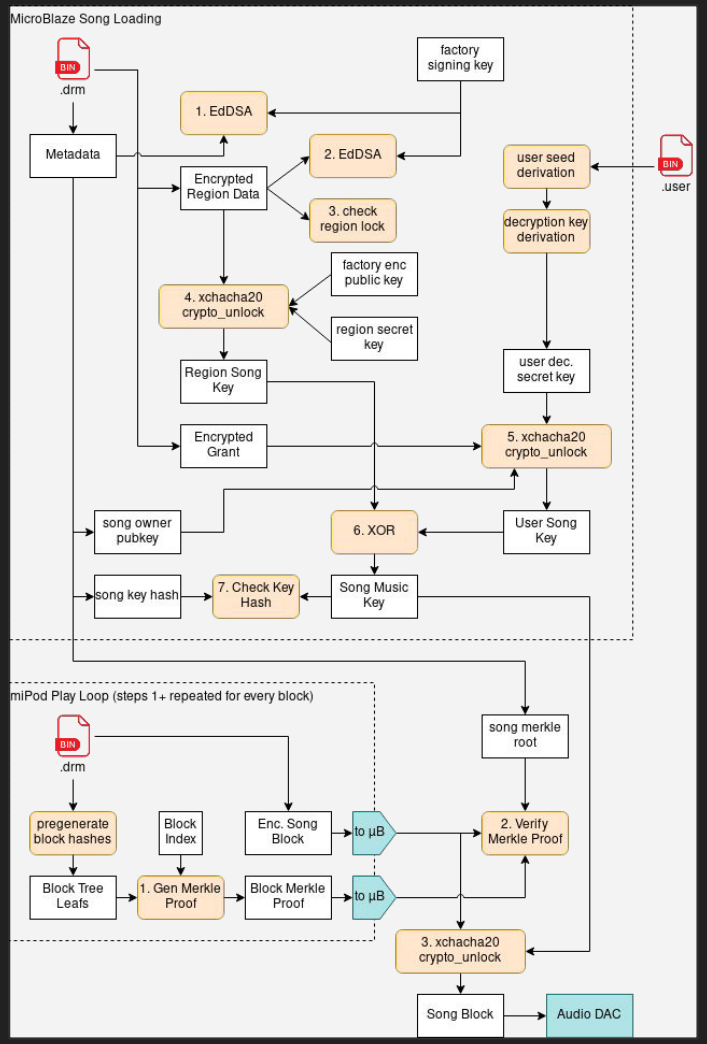
- ARM binary
  - easy to extract
- MB binary
  - difficult to extract, but could be possible

- Embedded in FPGA fabric
  - Good luck!!



FPGA fabric secrets used to derive user keys using kingcrab FPGA module

Flowcharts 2



Song playback

Goals: correct authorization (valid user, valid region), prevent tamper, prevent custom music

- Factory signature verification on metadata
  - Factory signing secrets never leave provisioning
- Owner keys authorize playback (for owner and shared users)
  - Deriving the user song key is a signed key exchange with the owner
- Region key stored on MB
- Song key = (user song key) ^ (region key)
- Merkle tree used for tamper protection

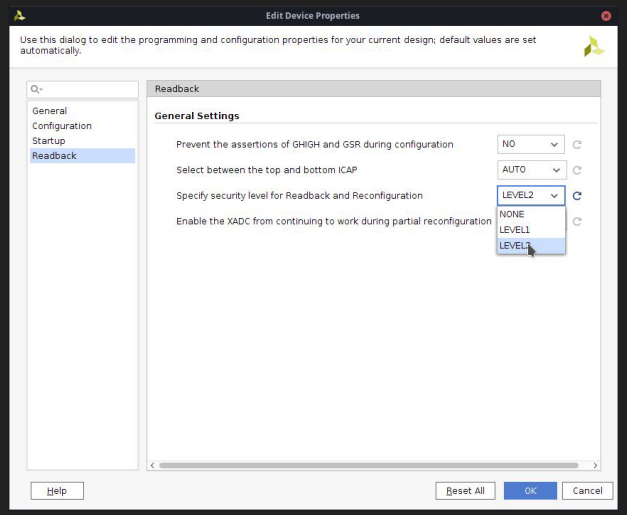
Additional Protections

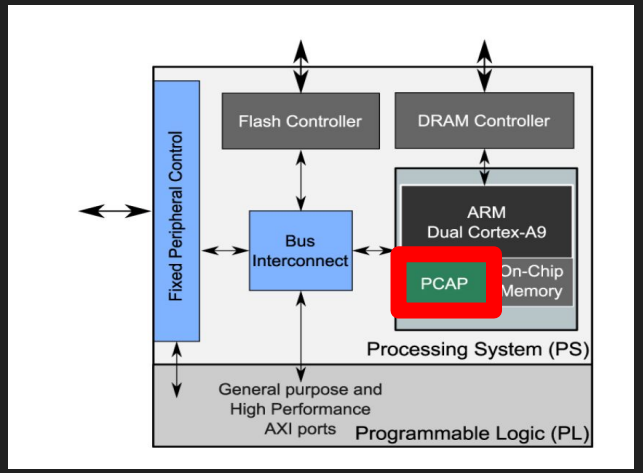
- State-of-the-art Data and execution Access 🅱revention (DAB) mechanism
  - Immediate wipe of sensitive buffers, then MB reset
  - 4 second (glitch-protected) delay in main()
- Stack canaries, branch glitching checks -> DAB
- All MB exceptions enabled -> DAB
- Invalid signature, invalid merkle proof, wrong PIN, … -> DAB
  - The official miPod player performs checks to prevent DAB
  - Custom player: good luck!
- XADC module: voltage and temperature alarms
  - Any alarm triggers reset
- Remove MB interrupt
  - No way to disturb secure DRM operations

Attacks

Flags / Teams	Husky Records	Competitor 1	Competitor 2	Competitor 3	Competitor 4	Competitor 5	Competitor 6
Custom Music	🏴‍☠️ Freepod	🏴‍☠️ Freepod	🏴‍☠️ Freepod	🏴‍☠️ Freepod	🏴‍☠️ Freepod	🏴‍☠️ Freepod	🏴‍☠️ Freepod
Music Tamper	💀 bug in digital_out	🐘 truncated file	📖 readback	🗑️ trashed file	🗑️ trashed file	🗑️ trashed file	🗑️ trashed file
Pin Extraction	???	🖥️ online bruteforce	📖 readback	📖 readback	📖 readback	📖 readback	🖥️ online bruteforce
Region Lock	⚠️ suspected	📖 readback	🏎️ race condition	🔀 swapped headers	📖 readback	🏎️ race condition	📖 readback
Unauthorized Play	???	☑️ possibly brute forceable no time :(	🏎️ race condition	🔀 swapped headers	📖 readback	🏎️ race condition	📖 readback

FPGA Attacks





- Bitstream security level prevents readback via USB/JTAG interface
  - MITRE boards can enable JTAG with PS register
  - Bitstream readback is blocked :(
  - Reconfiguration is blocked :(
- Does not apply to PCAP
  - Bridge between PS and PL configuration, used by FSBL for initial programming
  - Readback allowed!
- PL can be reset from the PS via register
  - New bitstream can be loaded

XDCFG_CTRL_PCFG_PROG_B_MASK (PCFG_PROG_B)	30	rw	0x0	Program Signal used to reset the PL. It acts as the PROG_B signal in the PL.

Freepod (arrrrr!)

- Reset FPGA using PS reset register
- Program super insecure example bitstream
- Play custom music like a true pirate

Impact

✔️ Custom Music

Mitigation

- Use the STARTUPE2 block in FPGA design to ignore resets

```
root@Cora-Z7-07S:~/music/foo# ./freepod-plus
jailbreaking...
MB> HAAX DRM Module has Booted
mP> Command channel open at 0xb4e35000 (33554564B)
MB> Queried player (2 regions, 4 users)
mP> Regions: eastcoast, southcoast
mP> Authorized users: ludacris, nas, biggie, gza
```

Bitstream Readback

PCAP can be used to read back secure bitstreams

Program the readback on non-MITRE boards (faster online brute force, easy)

Extract MB binary from BRAM and extract secrets (hard)

Challenges

- PCAP is poorly documented
- Bitstream format is poorly documented
- No existing readback example code
- BRAM is randomly ordered & scrambled

Bitstream Readback Steps

- Use our bare metal PS tool to read back PL and dump to SD card
  - Our friend uEnv.txt from last year is helpful for deploying tool over multiple designs
  - Dumps each design in ~1 second
- Use *SymbiFlow Project X-Ray* BRAM database to decode bits from dump
  - Obtain 100 bit sequences which are potential BRAM elements
  - 32 of the bit sequences form the bits of each word in the original MB code
- Brute-force bit sequence order to assemble binary
  - Use statistical analysis to guess order which best matches our MB builds for designs
  - ~3200 iterations, completes in seconds
- Extract secrets from reassembled binary

SymbiFlow

This would have been basically impossible without the SymbiFlow data — check it out at https://github.com/SymbiFlow/prjxray

Bitstream Readback (all your secrets are belong to us)

Depending on implementation

✔️ Region Lock
✔️ Unauthorized Play
✔️ Pin Extraction
✔️ Music Tamper

Mitigations

- Store secrets embedded in PL fabric
  - We are not convinced it's possible to extract these with any current tooling in < 60 days
- Design crypto resistant to secrets leaks
  - This was the one flag we ran out of time to get :(

Other Attacks

```
MB> Incorrect pin for user 'biggie'

MB> Incorrect pin for user 'biggie'

MB> Incorrect pin for user 'biggie'

MB> Logged in for user 'biggie'
mP> Successfully cracked pin for user biggie: 74291974
mP> Brute forcing complete; took: 527.542962.
miPod # ^C
root@Cora-Z7-07S:~/music#
```

- Music Tamper — trashing and truncating songs was very successful
  - Mitigation: make sure to do full integrity verification over *all* song data
  - Make sure each song is unique, swapping headers should be invalid
- Online pin brute force (*look ma no readback!*)
  - Mitigation: add delays to login process; ensure online brute force won't complete in time
  - The PS can trigger MB resets, make sure delays are persistent
- Race conditions on integrity checks
  - The DDR is shared memory, and can be changed by the PS during MB processing
  - Hash checks and authenticated encryption bypassed by changing the data after the integrity check on the MB
  - Mitigation: ensure all important data is copied to BRAM before integrity checks

Husky Records 2.0

- Fixing our bugs
- Add additional FPGA hardware-secret-based derivation to everything
  - User login
  - Song playback
  - Sharing
- Hardware-based glitch detection
  - PicoBlaze cores

General Comments

What design elements made things difficult for you as an attacker?

A: any element that made readback mandatory vs some easier attack

solid login rate limiting, proper user secret derivation

MB code with no standard vulns (race conditions, memory errors, …)

What are two pieces of advice that you would give to future eCTF participants?

A: read the manual, it's got some cool info in it

assume your design is vulnerable, and layer defenses to protect it anyway

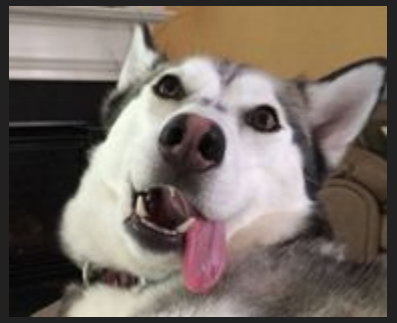
What would you do differently if you had to participate in this same competition again?

A: store more secrets in PL fabric

and actually finish the design before the attack phase starts :)

That’s all folks!

Any questions?



“I keep face-palming here. Our design is such that we have a big unobtanium door with like 10 locks on it taped into the doorframe and with all the windows open”
-- Cameron