

SSP21 BitW in FPGA

... because SCADA people don't patch!

Dr. Andrew Zonenberg





SSP21 refresher

- Developed by California utilities + LLNL
- Lightweight security for embedded systems
 - Much less bloat than TLS
- Designed for easy “bump in the wire” retrofits
- Multiple S4 talks from 2016 onward
- <https://ssp21.github.io/>



There's just one problem...

- Embedded Linux proxy platform likely has bugs
- We all know how good SCADA people are at patching
- Fixing that would be nice, but unrealistic



But what if we don't need patches?

- Provision once
- Deploy
- Never touch it again!



Sounds great, but how?

- 100% bug-free code is implausible
- But we don't need to fix ALL bugs for security
- Can we eliminate the dangerous bugs?



Engineering a robust solution

- Things we care about:
 - Protocol specification bugs (POODLE etc)
 - Crypto algorithm flaws
 - Code execution
 - Leakage of key material (Heartbleed etc)
 - Leakage of plaintext (if encrypting)



Protocol specification bugs

- No way to avoid patching if one is found
- Best defense is to minimize complexity
- SSP21 trims a lot of fat vs other protocols
 - No cipher suite negotiation
 - No ASN.1



Crypto algorithm bugs

- Best defense here is to use solid, well reviewed crypto
- SSP21 does this already
 - Curve25519
 - SHA256
 - HMAC
 - HKDF
 - AES-GCM*

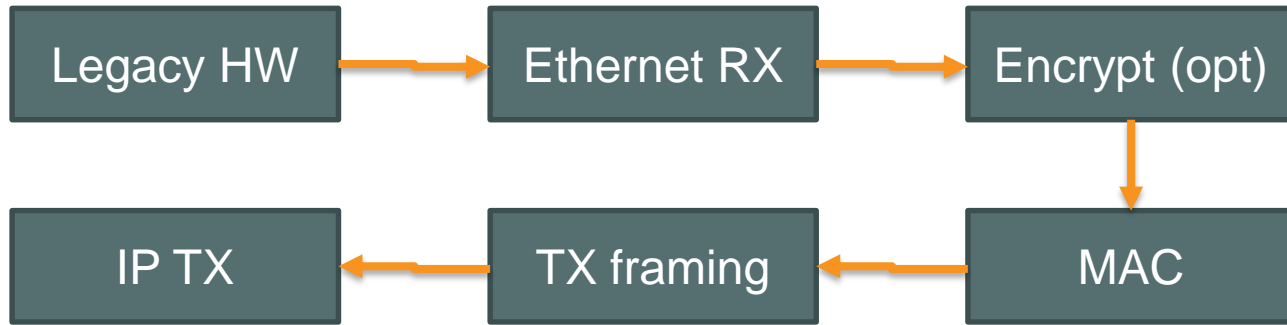


Code execution

- Trivial to prevent by construction
- If you have no CPU, no code can execute 😊
- All protocol logic is hard-wired state machines



Pipelined architecture





Key material leakage

- Also can be prevented by construction
 - Key storage is write only, can't be directly read
 - Only used by hardware crypto engines
 - No direct path from key to output



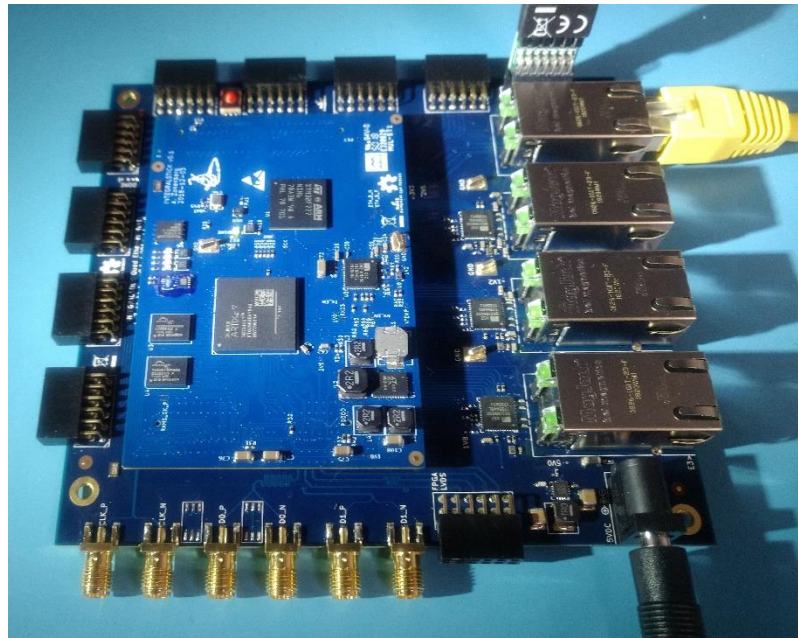
Plaintext leakage

- Bad pointers, etc are usually the risk
 - Memory-safe languages are a good start
 - But can you do 100% of your design in one?
- Pipeline-based architecture is inherently safe
 - Physically separate SRAM FIFOs for each stage
 - Address corruption can't lead to data jumping between buffers
 - Only path from RED to BLACK interface is through crypto logic



Development platform

- INTEGRALSTICK SoM
- Artix-7 FPGA
- STM32F7 MCU
- 4x 10/100/1000 Ethernet
- Other I/O unused





Crypto library

- Rolling your own crypto is normally a bad idea, but...
 - No de-facto-standard library of FPGA crypto primitives!
- Created a new open source crypto lib for this project
 - Permissively licensed (3-clause BSD)
 - Portable SystemVerilog
 - Cycle-accurate deterministic timing
 - Assumes endpoint is physically secure, not DPA-hardened
- 3rd party reviews welcome!



Crypto library

- Current primitives:
 - X25519
 - HMAC
 - SHA-256
 - Fortuna-based RNG/entropy pool
- Pending:
 - AES-GCM



VPN protocol

- Very simple layer-2 VPN framing
- Single header byte per SSP21 msg, then data
 - 0: Ethernet frame \leq 1000 bytes
 - 1: First 1000 bytes of larger Ethernet frame
 - 2: Remaining bytes of larger Ethernet frame
- No jumbo frame support



VPN protocol

- Running at layer 2 makes the VPN transparent
- No special endpoint config required
- DECnet or BACnet? Doesn't matter 😊



Implementation status

- SSP21 handshake over UDP completes successfully
 - Currently only pre-shared symmetric key supported
 - X25519 code for pre-shared pubkey is done, but not integrated with protocol stack yet
- VPN is still a WIP
- Currently uses 30% of XC7A100T
 - Not optimized, has a lot of debug logic



Questions?

- Contact: Andrew.Zonenberg@ioactive.com
- Code:
 - <https://github.com/azonenberg/ssp21-vpn>
 - <https://github.com/azonenberg/antikernel-ipcores/tree/master/crypt>
 - <https://github.com/azonenberg/antikernel-ipcores/tree/master/protocol/ssp21>