galois

DEF CON 2019 Voting Demonstrator

The BESSPIN Voting Demonstrator Team

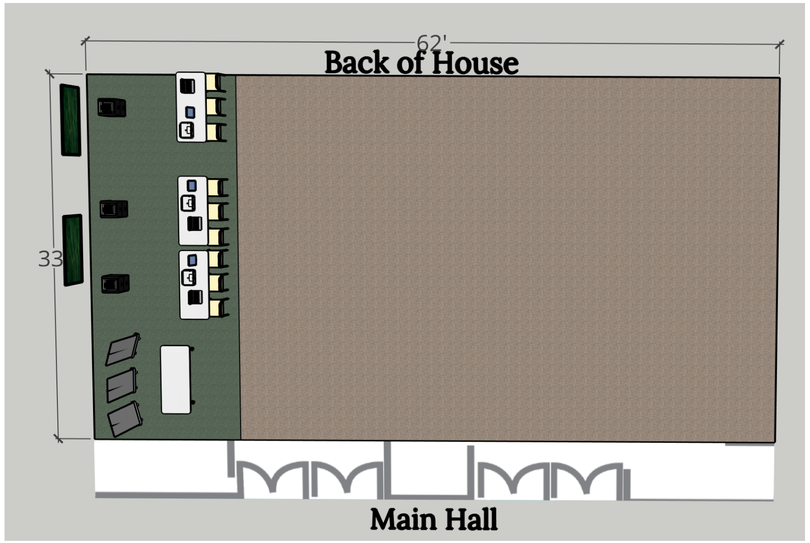
August 2019

2019 Voting Demonstrator — Details

The 2019 DEF CON demonstrator has 3 components:

- Smart Ballot Box (SBB) — SSITH
  - 32-bit RISC-V microcontroller and ballot box hardware (buttons, lights, sensors, paper feed via a single motor, real-time clock, SD card)
  - FreeRTOS, including TCP/IP stack and drivers for ballot box hardware
  - secure logging (hash-chained logs with signed entries) to both SD card and network
- Ballot Marking Device (BMD) — non-SSITH
  - Intel NUC running a TypeScript application in Chrome browser
  - touchscreen display with which voters interact
  - standard printer (small Brother laser) for ballots
- Reporting Server (RS) — non-SSITH
  - Intel NUC running an HTTP server and a small software layer (currently implemented in Python) to receive and validate log information from SBB and tally the election

2019 Voting Demonstrator — DEF CON Layout



- voting demonstrator space is part of the larger DEF CON voting village
- the big screens are information displays showing both static information (about the SSITH program, the voting demonstrator, the point of the exercise, etc.) and dynamic information (about the running elections, exploits that have been successful, etc.)
- each micro controller being demonstrated has its own “voting system” (SBB, BMD, printer) as depicted here; RS is at the “admin table” (not depicted here)

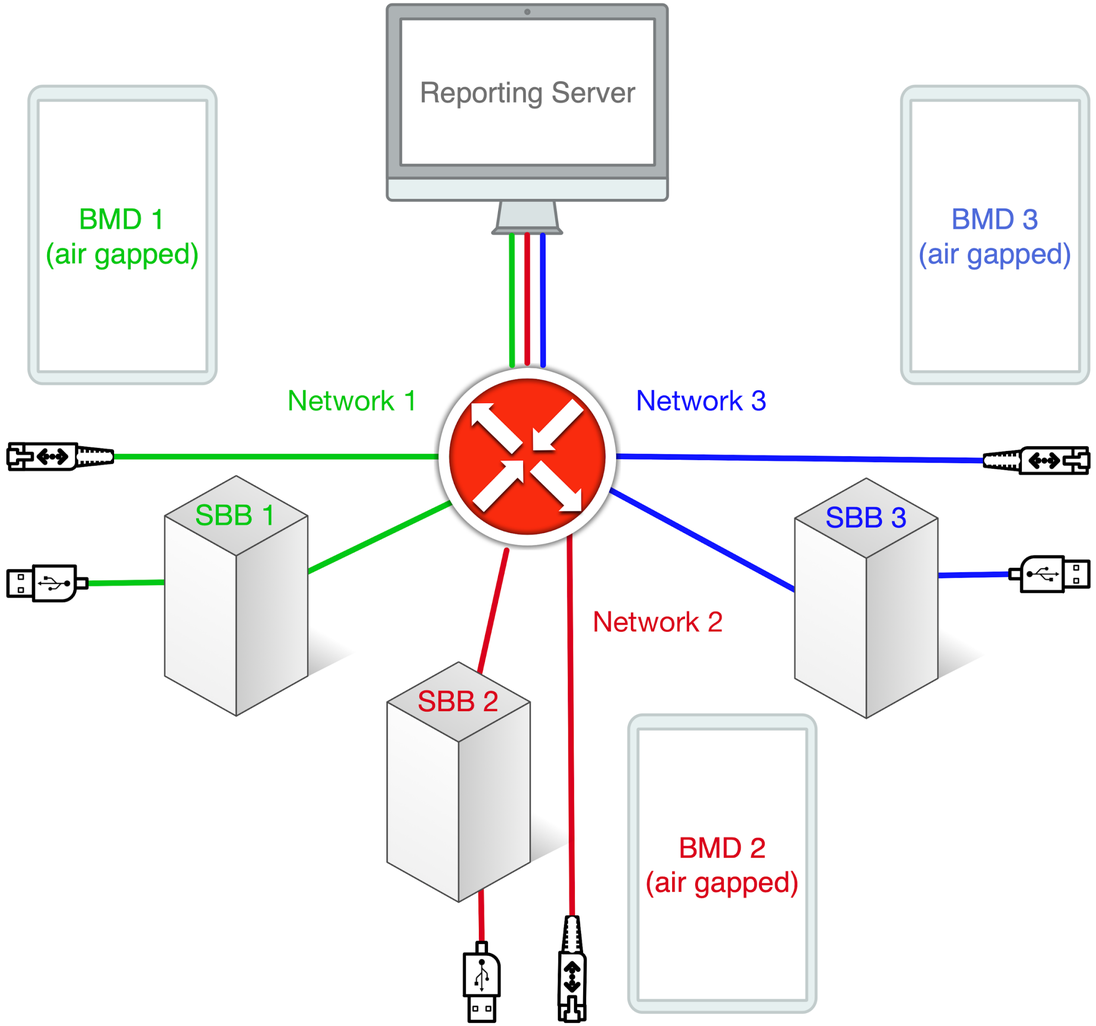
2019 Voting Demonstrator — DEF CON Layout

- each micro controller being demonstrated has its own work table, at which there will be at least one Ethernet drop and a single USB UART connection available
  - network details on the next slide
  - nothing will be connected to Internet from the floor of DEF CON (this year)

- red teamers will connect to Ethernet, optionally download the red team virtual machine image from the RS, and engage with the SBB as we will describe next
- there will be a GitHub repository — https://github.com/GaloisInc/SSITH-Voting-System-Demonstrator-2019 — that contains the source code for the SBB and a place for red teamers to file issues documenting their exploits

2019 Voting Demonstrator — Network Layout

- switch isolates the three SBBs (and their hackers) from each other while allowing the reporting server to gather logs from all SBBs
- BMDs are air gapped but are associated with specific SBBs by sharing ballot signing keys (more later)



2019 Voting Demonstrator — Election Flow

a voter participating in the demonstration election does the following:

- go to the BMD, make choices in the various races/questions, and print a ballot, following the instructions on the screen
- take the ballot to the corresponding SBB and feed it in
  - there is a timeout period (currently 10 minutes): if the voter walks away with the ballot and doesn't feed it into the SBB within the timeout period, it is considered invalid
- choose, when prompted by the SBB, to cast or spoil the ballot
  - casting a ballot means it will count in the election — it is consumed by the SBB
  - spoiling a ballot means it will not count in the election — it is returned to the voter by the SBB, and may not be used again
  - ballot spoiling is a critical component of the mechanism we will use in 2020 for end-to-end verifiability, but is not used for such in 2019
- repeat as desired (we are not making sure that individuals only vote once)

2019 Voting Demonstrator — Secret Information

each SBB has three AES keys in memory that are used to compute AES-CBC-MAC “signatures” and need to be protected:

- one, shared with the corresponding BMD, is used for barcodes
  - a BMD signs a barcode at printing, the signature is checked when the SBB scans the barcode, and an SBB will not accept a barcode from a different BMD (or an invalid one)
- two, shared with the reporting server, are used to sign log entries
  - one is used to sign the initial log entry contents, and the result is used as the initial “hash” for the root of the log’s hash chain
  - the other is used to sign subsequent log entry contents and computed hash chain hashes
  - two are required because it is insecure to use a single AES key for AES-CBC-MAC of different lengths of data
- these keys are compiled into the software of the SBB this year (there is no hardware security module), so there are specific regions of memory that must be protected

2019 Voting Demonstrator — Red Teaming

- to “win”, red team participants must exploit a *buffer overflow* or *information leakage* in order to cause the system to behave incorrectly
  - examples: accept (and count) a ballot that was not produced by the BMD; reject a ballot that was produced by the BMD and inserted into the SBB in a timely fashion; record as cast a ballot that was spoiled, or vice-versa; change the tally in a non-detectable way
- if the incorrect behavior would be detected by standard practices — e.g., if the tally is changed, but cryptographic inspection of the logs makes it obvious that they were tampered with — that’s not a win
- wanton destruction (stomping over memory arbitrarily to break the system, wiping the logs on disk, DOSing the reporting server) is not a win
- the most straightforward way to win is to exfiltrate the AES keys and use them to falsify barcodes (on printed ballots) or log entries (that are written to disk or network or both)

2019 Voting Demonstrator — Red Teaming

- we are providing red team participants with the following:
  - full source to the voting system software, including our FreeRTOS modifications and secure boot
  - bitstreams (but not source) for the systems in use at DEF CON
  - memory image of each SBB
  - live access to each SBB via Ethernet and UART (over USB)
  - VM image with cross-compilers, etc., to get them started quickly
  - example PoC exploits that will compromise an SBB on bare GFE
- we are also providing red teamers with a way to read arbitrary memory, and write to a specific memory region, using an on-device web server:
  - can “peek” at arbitrary memory addresses (on bare GFE, but sensitive areas of memory should be protected on SSITH hardware)
  - can “poke” new data to a specific region of memory and trigger execution to jump to the beginning of that region by making an HTTP request
  - malware is called as a C function, to enable stack smashing/ROP in a straightforward way

2020 Voting Demonstrator — Preview

the 2020 demonstrator will differ from the 2019 demonstrator as follows:

- *all* components (BMD, SBB, RS) will be on SSITH hardware
  - this probably means running a browser with JavaScript, connected via USB and video (HDMI or similar) to a touchscreen, on P2 or P3
- the system will be *end-to-end verifiable* — voters will be able to confirm that their ballots made it into the final tally and were counted as cast
  - some aspects of this already exist in 2019 but not in a way that preserves voter privacy/anonymity
- *all* components in a single “polling place” will be networked together
  - the end-to-end verifiability of the system requires the BMD, SBB, and RS to communicate in order to track printed/cast/spoiled ballots
  - the RS will serve more of a polling place coordinator role in 2020 — there will be one of them per polling place, not a single shared one
- there will be an *optical scanner* doing OCR (or similar) on *hand-marked paper ballots* in addition to the BMD
  - it will most likely be the same machine as the BMD, at which a voter can choose whether to scan a ballot or make a new ballot