Gmail Chris Jerdonek <chris.jerdonek@gmail.com>

Thursday Meeting

Carl Hage <osv@carlhage.com> Thu, Nov 16, 2017 at 11:52 AM
To: "Jerdonek, Chris (REG)" <chris.jerdonek@sfgov.org>

Hi Chris,

I'm working in Germany so won't be able to attend this month.

There are a few things I would have brought up. I'll note that here, and you can do what's appropriate, either mention it on my behalf or wait until the next meeting.

I noted in the minutes from last month mention of some notes on results formats and open data. I'd be happy to write up some notes on the NIST standards as well as other results formats. Also on a glossary, as applied to what we are working on.

Some things I found:

The California Secretary of State certified 2 remote "accessible" vote-by-mail systems for use in California.
http://elections.cdn.sos.ca.gov/ccrov/pdf/2017/october/17089sl.pdf

People with disabilities or military and overseas voters can use their own computers to prepare a ballot (using a web browser) and print a page. The normal computer paper is sent with a special envelope and the mail ballot return envelope. I believe the generic paper ballots are re-marked onto official ballots by election department staff. (The voter must agree to that.)

This approach might be adopted by SF to facilitate disability access in a vote-by-mail system.

Vendors approved are:
http://www.fivecedarsgroup.com/
http://democracylive.com/our-products/liveballot-online/
https://app.liveballot.com/v2/ballot

These are designed to be re-marked by people, but in general might be possible to use better ballot scanning software that could be used with letter-size paper made with personal printers and generic browsers.

Some new links:
OSET: http://votestream.trustthevote.org/
[Demo just covers election results.]

The original Open Voting Consortium paper-- just a demo but has a good outline of using COTS hardware for precinct voting:
"A PC-Based Open-Source Voting Machine with an Accessible Voter-Verifiable Paper Ballot"
http://gnosis.cx/publish/voting/electronic-voting-machine.pdf
[Usenix paper - 2004 demo to Santa Cruz ROV]

The Travis STAR-Vote produced a paper describing the approach:
http://www.traviscountyclerk.org/eclerk/content/images/presentations_articles/pdf_tc_elections_2013.07.26_star.pdf
This has a good decription of how COTS equipment can be used with precinct voting, including a detailed description of how crytography is used. The approach is also similar to the LA County

A new commercial (not open source) company using COTS hardware for precinct accessible voting:
Used with: https://www.clearballot.com/technology/clearaccess

https://www.youtube.com/watch?v=GG4oMstRbtk
https://www.verifiedvoting.org/resources/voting-equipment/clear-ballot-group/clearvote/

Clearballot also makes COTS hardware central scanners, and now has a semi-custom precinct scanner.
https://www.clearballot.com/technology

---
2 other open-source voting project/demos:

PVote: http://pvote.org/ A Python-based voting machine. Illustrates using open precomputed data to simplify the sensitive source code (460 lines of python). Had a detailed code review.

Votebox: http://votebox.cs.rice.edu/ Another example of using pre-rendered data. In my opinion, an overly complex challenge/verification process.

I've took a look at some of the prior open-source voting software. I could write some more lengthly comments if you wish.

In virtually all of what I found, none had much in the way of comments in the code, or in documentation. Even PVote with 90 hours of review scrutiny is without comments (I would have rejected it).

Prime-III is supposedly used by some election officials for disability-access, but it's still a highly demo state, and uses old fashioned somewhat clumsy javascript with old-style frames. The output, (Similar to Alternative Format Ballots), is a generic and somewhat simple output.

In spite of klunky code, Prime-III is a really good example at importing other apps packaged with the browser-run HTML and javascript. It includes third party open-source javascript to do voice synthesis (everything is synthesized-- no audio recording support), voice recognition, and the QR code generation. Emscripten is used to package open-source packages to run in javascript, and it's not really large. It also illustrates the LEVI (Low Error Voting Interface) a GUI ballot experimentally designed to reduce user errors and confusion.

I couldn't make much sense of the STAR-Vote demo implementation because of no comments and no documentation, and it's in the haskell programming language.

OpenCount is a giant monolithic application driven by a GUI. It has a huge amount of prerequisite library software so not so easy to install. This brings to mind some project requirements (see below).

One useful project architecture requirement is to separate the sensitive code (e.g. voting machine or tallying software) from data preparation, conversion, and audit/check/validation code. Care should be taken to minimize the size and prerequisites for sensitive code to make it easier to review and audit. The data preparation, conversions, and auditing, make a modular design. PVote is a good example of this principle. OpenCount is a good example of the wrong way-- it's just one giant blob, GUI-driven not command line, and has lots of prerequisite software.

I had a great deal of trouble installing and using some of the open source voting software I looked at because of a complex set of OS requirements and prerequisite software. OpenCount crashes in the OpenCV image processing libraries, maybe because of some version incompatibilities.

Although I recommend using open-source libraries, I think there should also be a goal to minimize the total amount of code including imported libraries-- especially for sensitive applications. For example, it might be better to copy-and-paste the few routines needed from some open-source library rather than import everything. Also it is good to avoid libraries with a huge chain of other library dependencies.

The Secretary of State requires certified equipment to be submitted with all software included. So that means the operating system, and build software, as well as library software along with source code. I think an open-source repository should have copies/snapshots of library software, and well as the build environment. I think a good goal would be that any outside person could replicate the build process to validate (do a binary compare) on digitally signed firmware, boot-images, and runtime images. So that means capturing the development environment as well.

Probably containers (e.g. Docker) would be a good approach, both to setup, update, and capture the development environment with source code, and also to create secure runtime environments (also with secure boot and secure

hypervisor). The Docker system can be used to create base images using public repositories of standard OS files and software development tools. In theory, third parties could replicate Docker images and/or use downloaded containers to replicate the build and runtime process. It makes a better controlled update process.

It also might be worth having a standard virtual machine image to run the containers in, e.g. a VirtualBox image that could be used on a windows/macos/linux machine. Ultimately, it should run on a secure boot machine with small base-OS configured to run secure-signed container apps.

I haven't thought this all out-- just some ideas for now.

A topic for future discussion might be preparation of a decision tree or something similar-- essentially listing the options for making ballots, voting, kind of equipment, counting, etc., and some comments on pros & cons with each alternative.

For example, are all ballots the usual hand-marked paper same as used for vote by mail, so ballot marking devices must print on mail-ballot paper.

For precinct voting, accessible voting devices need an audio interface and ballot marking device. Should it print an alternate format, e.g. single page summary of selected choices or mail ballot circles/arrows? If machines are available for disability access, can they be used for voting by everyone? If you have a voting machine (vs pen), then it can record digitally signed cast-vote-records, so better security. It could also read a QR code to allow votes to be prepared at home or on a phone, then scanned to pre-load into the voting machine for printing on an official ballot-- recorded electronically and centrally scanned for audits.

Then there are options like precinct vs central-only scanners for pen-marked ballots.

Then there are hardware issues, both precinct and central.

We might need to mention software patents. Something we might want to do might be covered by a patent. [The OpenCount developers have some patents, for example, related to ballot scanning.]