Voting Machine Technology

Tom Trumpbour
Computer Software Consultant
United States

History of Voting Machines
Uniform Paper Ballots

- Voters make choices in private by marking boxes, then dropping ballot in sealed box
- First uniform paper ballots were used in Victoria Australia in 1856.
- Were used in the US for the first time in the State of New York in 1889.
- Currently less than 2 percent of voting in the US

History of Voting Machines
Mechanical Lever Machines

- Lever closes privacy curtain which initializes for voter; horizontal levers are turned which are finalized when opening privacy curtain with lever. Counts with dials which move 1/10th for each digit.
- First used in Lockport, New York in 1892
- Every major US city used by 1930's
- Over half of US votes by 1960's; now 20%

History of Voting Machines
Punchcards

- Voters punch holes in cards which are put in ballot box and later computer tabulated at precinct.
- First used in Georgia in 1964
- Two types: votematic (names correspond to numbers) and datavote (names next to holes to be punched)
- In 1996 used by estimated 37 percent of US

History of Voting Machines
Optical Scan

- Much like tests taken with number 2 pencils where one blackens the choice and is later tabulated by a machine
- Began use in the early 1980's but exact origin unknown
- Used by about 24 percent of US by late 1990's

History of Voting Machines
Direct Recording Electronic (DRE)

- Like lever machines, most do not currently rely on paper. End user touches screen or push buttons. Stores results to memory, disk or smart card.
- Began use in the 1970's
- By late 1990's used by about 8 percent of the US. Today over 25 percent.
- Bulk of this talk will concentrate on DRE's

Voting Machine Reliability Statistics

Residual vote for machines from 1988-2000

- Optically Scanned 1.3%
- Lever Machines 1.4%
- Paper Ballots 1.5%
- Punch Card 2.5%
- DRE 2.7%

Diebold
Code on Public FTP Site

- Code was discovered on Diebold's ftp site January 2003
- While Diebold officials seemed un-concerned about proprietary code on the Internet, they took it off upon discovery
- Diebold has not taken action about mirrored code available publicly

Code Highlights

- The code is for the terminal voting machines, AccuVote-TS and not the GEMS back-end management system
- Code was written in C++, which if not in tight control, is vulnerable to buffer overflow attacks
- The terminals are configured under several Microsoft Windows environments (i.e. 95)

Design Summary

- The ballot is configured and stored to a file which can be installed by removable media or via an outside connection
- System is initialized and voters vote with smart cards, to be reset after voting
- Ender or administrator smart card gives ok to send results to be tallied at GEMS

Vulnerabilities

- CE mounting drive as directory-take out drive and just create directory
- One can program a smart card-as multiple voters or for administrative functions
- Encryption not used or used properly. Reporting to GEMS does not use any encryption

Vulnerabilities

- Phone lines, Internet, wireless all ways for man-in-the-middle attacks
- No change control process to prevent coders from adding malicious code
- Two methods for DoS attacks: Admin or ender card and ballot definition from remote
- Pin numbers are unprotected, stored on card
- Password and key are hard coded in source

Vulnerabilities

- Fail back to use manufacturer's default password on smart card
- Configuration is done in the clear within the registry; i.e. terminal serial and COM port
- Candidate order on ballot matters
- The database is MS Access; not very robust like SQL Server or Oracle
- Database design lacks referential integrity

Vulnerabilities
Code Samples

```
SMC_ERROR CCLXSmartCard::Open(CCardReader* pReader)
{
... [removed code] ...
// Now initiate access to the card
// If failed to access the file then have unknown card
if (SelectFile(0x3d40) != SMC_OK)
    st = SMC_UNKNOWNCARD;
// Else if our password works then all done
else if (Verify(0x80, 8, {0xed, 0x0a, 0xed, 0x0a, 0xed, 0x0a,
   0xed, 0x0a})
    == SMC_OK)
    st = SMC_OK;
// Else if manufactures password works then try to change
   password
else if(Verify(0x80, 8, {0x01, 0x02, 0x03, 0x04, 0x05, 0x06,
   0x07, 0x08})
    == SMC_OK) {
    st = ChangeCode(8, {0x01, 0x02, 0x03, 0x04, 0x05, 0x06,
   0x07, 0x08},
                                    8, {0xed, 0x0a, 0xed, 0x0a,
   0xed, 0x0a, 0xed, 0x0a});
    // Else have a bad card
```

Vulnerabilities
Code Samples

Audit Log Encryption:

```
        #define DESKEY
((des_key*)"F2654hD4")


        DesCBCEncrypt((des_c_block*)tmp,
(des_c_block*)
                record.m_Data, totalSize,
DESKEY, NULL,
                DES_ENCRYPT);
```

Vulnerablilites
Code Comments

```
/*
 *   Parse the file until the EOF Token.  If we get too many errors
 *   at this level, we abort.
 */

/* A simple token is anything that starts with a character that
 *   cannot be part of an identifier.  Here we do a brute force
 *   search of every sub-string starting with this character up
 *   to 9 characters in length.
 * XXX Why the magic 9???
 */
```

Vulnerablilites
Code Comments

/* XXX Okay, I don't like this one bit. Its really tough to tell where m_AudioPlayer should live. CBallotWnd is not bad, since that is where it is used most, and CBallotWnd is always around when you need to play audio. Its also the only window that is (currently) interested in OnAudioFinished messages.

Except it seems, the CLanuageSelDlg. The solution chosen is to construct the CBallotWnd early and fire up VIBS with InitAudio(). Its probably fair to ask why CInstructionDlg is a child of CBallotDlg but not CLanguageSelDlg. CBallotDlg might
be a better place for m_AudioPlayer. A reorganization might be in order here.
*/

{ /* XXX ERROR ERROR ERROR */

Web Resources

- http://www.countthevote.org/temptsx/cvs.tar
- http://www.blackboxvoting.org
- http://www.eff.org/Activism/E-voting/
- http://www.wired.com/news/evote