Trojan Horse in DRE
Application Software

Method

Malicious code hidden in DRE polling station that enables:

1. Someone to access special features that can siphon votes from one candidate (or option for questions) to another after ballot definitions have been determined.
2. Votes to be siphoned between candidates based on predetermined criteria such as moving votes between candidates associated with political parties. This would require the DRE software to read the ballot definition files. Votes could also be siphoned between candidates based on non-party-based attributes such as percentage of the vote received.
3. The attack could also result in disrupting an election, perhaps via a denial or service type method.

Access to each DRE, the host server(s) or machine(s) on which the master copy of the source code, or compiled binary image(s) of the application software are created and/or stored, or any intermediary system(s) that might be responsible for installing software onto the DRE's.

Taxonomy

Wholesale, Configuration-related, Change Management, programming, software.

Applicability

DRE, DRE with VVPT

Resource Requirements

For any of the three attack methods above, the perpetrator must be a skilled programmer and have access to the source code. This could occur at either the vendor (or a sub-contractor) site, or a test lab (assuming the vendor has provided the source code to the VSTL). It could happen within an elections office, but only in the case where the vendor made source code and installation procedures available at the local facility, which is not a vendor's typical method of operation. The perpetrator must be skilled at understanding how votes are actually created and tabulated, the methods used for internal auditing and crosschecks, and how the code is tested.

This copy from Black Box Voting document archive - http://www.bbvforums.org/forums/messages/2197/2400.html

For a Type 1 attack, there must also be a perpetrator with access to the DREs at the states and counties. The perpetrator does not even need access to all counties, or even all precincts, but can could be effective by targeting DREs to be placed within critical demographic regions. The access can be either authorized (an election official, e.g., whose job it is to input ballot definitions, test DREs, or otherwise touch a large number of machines) or unauthorized (e.g., via a break-in to a storage warehouse).

Potential Gain

The potential gain is large. Since the attack impacts many separate DREs, a small number of votes can be stolen per DRE adding up to enough votes to change the results for a race.

Likelihood of Detection

The malicious code could be detected in several places: by the vendor, by the test lab, or by an election official noticing anomalous results during a test or in a real election. A skilled programmer will generally be able to hide a significant amount of dangerous code without being detected in testing. (See countermeasures.) Detection would depend on the individual skills and depth of the source code review (either at the vendor or the test lab), or the amount of attention being paid to each DRE's behavior during testing or in a real election.

Countermeasures

Source code review: User interface code, for example, tends to be extremely complicated calling multiple libraries. Source inspections and reviews that might catch this type of code typically cost over $500,000 and take over 6 months. In addition, any change to the source code must result in a similarly expensive re-review.

> Open-ended testing: This testing also is very expensive and requires significant security analysis expertise.
>
> Testing that fully simulates Election Day activities
>
> Independent Dual Verification with audit
>
> Parallel Testing

Citations

Ken Thompson, Turing Award Speech, 1984: http://cm.bell-labs.com/who/ken/trust.html

This copy from Black Box Voting document archive - http://www.bbvforums.org/forums/messages/2197/2400.html

The moral is obvious. You can't trust code that you did not totally create yourself. (Especially code from companies that employ people like me.) No amount of source-level verification or scrutiny will protect you from using untrusted code. In demonstrating the possibility of this kind of attack, I picked on the C compiler. I could have picked on any program-handling program such as an assembler, a loader, or even hardware microcode. As the level of program gets lower, these bugs will be harder and harder to detect. A well installed microcode bug will be almost impossible to detect.

Retrospective

While there is no evidence that wholesale vote fraud has occurred using DREs, the issue is whether this is possible in the future. Given the large payoff possible, the relatively low likelihood of detection if a very skilled programmer in involved, the large number of very skilled programmers available, and the small number of perpetrators necessary, this threat is a serious threat to consider for the future.

This copy from Black Box Voting document archive - http://www.bbvforums.org/forums/messages/2197/2400.html