Software Independence

Alec Yasinsac
Co-Director, Security and Assurance in Information Technology Laboratory
Florida State University
Tallahassee, Florida 32306-4530

December 10, 2007

Abstract

Software independence describes a voting system architecture that offers to standardize voting systems through the certification process. This architecture ensures natural redundancy in electronic voting systems. In this paper, we offer several observations and open questions regarding security properties and prospective side effects.

We see a critical omission in the present VVSG draft relating to development processes. It is our contention that all voting systems must be engineered for high assurance accomplished through rigorous process maturity management.

1. Introduction

In August 2007, the Technical Guidelines Development Committee submitted recommended guidelines regarding the Voluntary Voting Systems Guidelines (VVSG) [1] (hereafter referred to as the VVSG draft) to the Election Assistance Commission. This draft proposes to incorporate the emerging concept of software independence as a requirement for all voting systems that are submitted for certification.

1.1. Software Independence Defined

Software Independence (SI) is a self-describing phrase that attempts to capture the notion of preventing reliance on computer programs in voting systems. The definition of SI in the VVSG draft is that:

> ... an undetected error or fault in the voting system's software SHALL NOT be capable of causing an undetectable change in election results.

1.2. The Imminent Results

If the VVSG draft is adopted with the SI concepts intact as they now appear, all voting systems certified under this standard and for the foreseeable future, will be required to incorporate paper trails and a rigorous election-day audit system.

2. Independent Mechanisms

The core trust property of software independence is that it provides redundancy, which is a well-understood approach for developing reliable, robust, and secure systems. Prior to SI, recounts became the de facto standard voting system redundancy approach and is widely adopted, albeit with some significant variations in implementation details. SI expands this notion by adding precision to the necessary nature of the employed redundant mechanisms. This is an important advance.

As an analogy, consider skydivers, most of whom carry a primary parachute and a backup parachute. These parachutes may be identical, so their "independence" is simply that they are deployed at different times, so if the main parachute fails due to a flaw in workmanship or damage

after deployment, the backup parachute likely provides sufficient independence. On the other hand, if the main parachute fails because of environmental conditions such as lightening or wind, it is uncomfortably likely that the backup parachute may fail as well.

The corresponding "independent mechanisms" that software independence present are electronic ballots and paper ballots, with electronic systems corresponding to the main parachute and paper ballots corresponding to the backup parachute. These mechanisms differ in many ways that can facilitate verification. There are questions about whether or not the paper-based system requirement is overly restrictive. Conversely, the focus on "software" independence may leave voting systems vulnerable to devastating, post-voting-period attacks.

An important question that must be answered before adopting software independence as an across the board requirement is:

> Is it possible to efficiently generate multiple independent electronic records that can verify one-another?

If so, the standard should require "independent redundant mechanisms" rather than "software independence".

Let's go back to the skydiver example. The backup [parachute] system is similar (or identical) to the primary system. Thus, they are not very independent, except that they are different parachutes that are released at different times. However, they are the same technology that may be affected by the same environmental factors (wind, rain, lightening, etc.), the same design flaws, and the same attacks (an attacker could damage both with a knife in the warehouse), etc. Also, the primary parachute may interfere with the backup parachute if the primary is not released from the pack when the backup is triggered. So in this case, there is both physical dependence and architectural dependence between the main and backup system.

In system verification/redundancy, the more independent the mechanisms, the stronger the verification they can provide. So, the best situation is to have two totally independent mechanisms, which begs the question: Why not have a more independent backup parachute system? The answer is that for skydivers, other factors dominate, e.g. size, weight, cost, reliability, etc. Similarly, we must assess if composing a primary system based on electronic ballots with a secondary system based on paper ballots is necessary (i.e., could a less restrictive architecture accomplish an equivalent trust level) and sufficient (i.e., does

2.1. How Much Independence is Necessary?

Analogously, software independence uniformly promotes paper's security properties over electronic redundancy. In order to determine if software independence is appropriate, several questions regarding general voting system redundancy for composed systems (s1 = x1 + y1) should be answered:

1. Are there differences between x1 and y1 relative to s1's security properties?
2. Do the differences limit (or enhance) s1's functional capabilities?
3. What are the security properties of the individual mechanisms themselves?
4. What security properties are retained when component's are composed?

To date, we have not answered any of these for present VVPR systems with any authority.

2.2. Paper Trail Security Properties

We cannot forget that the push to renovate and improve our election systems results from a lengthy history of election problems based on the limited security properties that are inherently in paper-based systems, e.g.:

- ID cards, birth certificates, etc. are easy to create and reproduce
- The foundational technology that provides a modicum of protection for these documents (bonding, watermarking, etc.) does not scale to voting systems, so ballots will necessarily have lower inherent security properties than birth certificates, etc.
- Hanging chads, butterfly ballots, and lost and reappearing boxes of ballots are recent examples of paper ballot's accuracy and reliability frailties

Voting system attackers operate with limited information during the voting period. For example, victors are not known during the voting day and there is little value in manipulating a contest that will be won anyway. Similarly, victory margin cannot be predicted and it is not useful to manipulate votes and still loose the target contest.

On the other hand, it is well-known that paper ballots are vulnerable to malice after the voting period ends, when the outcome is clear, the margin is known, and malicious attacks can be precisely honed to accomplish the desired intent.

The first question we must address if we decide to require future voting system technology to be paper-based is:

- What are the inherent security properties in paper used as Voter Verifiable Paper Records (VVPR)?

An equally compelling question is:

- How does software independence solve or reduce the problems with paper ballots that drove us to where we are?

It is not clear how the Technical Guidelines Development Committee (TGDC) -proposed VVSG standard mitigates the wide array of problems in paper systems. If it does nothing, we are destined to see the next generation of hanging chads, butterfly ballots, and lost ballot boxes.

2.3. Retail and Wholesale Impact

One profound impact that electronic voting systems introduce into election systems is the opportunity for a few individuals (or a single individual) to maliciously control a large number of votes. This phenomena is termed "wholesale attacks", reflecting the potential to have broad impact, for example, during system manufacture. Thus, more traditional attacks, such as ballot box stuffing and ballot tampering correlate to point of sale/vote, or retail, attacks.

2.3.1. Wholesale and Retail Attacks, Paper and Electronic Ballots

The property of paper systems that has carried the day on the paper trail argument is that paper-based systems are not susceptible, or are highly resistant, to wholesale attacks during the voting period. This is good.

Conversely, an issue that has been largely ignored in this debate is that electronic voting systems can prevent (or at least strongly limit) retail attacks after the voting period ends. One relative architectural protection for electronic ballots is that after voting ends, there are no processes that move data into an electronic voting terminal; rather, the processes are designed to move data outward from the terminal.

Additionally, by its nature, it is difficult to selectively modify electronic votes after the results are reported, leading to the well-founded, common argument that electronic ballot recounts are little more than regurgitations of the original count. Fortunately, electronic tallies can be mechanically precise, protecting electronic results is well understood, and electronic integrity protection computations are inexpensive. Thus, electronic ballot recounts should rarely vary from the original count. This is also good.

So the most important question related to a decision about whether software independence or independent mechanism is a more appropriate standard is:

> Is it possible to engineer a voting system that composes two electronic systems so that their independence prevents wholesale voting fraud?

Unless the answer is definitively "no", verification theory suggests that independent redundant mechanisms, not software independence, should be the voting system standard.

2.3.2. The Official Record

While redundancy in a decision process can add confidence through verifiability, having redundant processes naturally begs the question of how to resolve conflicts when the mechanisms differ. Some argue that a paper record, whether voter or machine marked, should be the vote of record. Conversely, when paper ballots are damaged or lost, electronic ballots may provide the only record of some votes.

The observations in Section 2.3.1 above suggests an important role distinction that can strengthen the verification that these mechanisms allow. Since paper ballots' properties naturally deter wholesale fraud, paper ballots should be used to audit electronic mechanisms to detect faults or attacks during the voting period. When anomalies are detected, elections officials can analyze all information to reconcile the count, relying on the paper ballot as official in the first tally.

Conversely, once the first tally is verified and reported, official ballot status should shift to the electronic record that is naturally resistant to retail attacks after the voting period ends. Again, elections officials must consider all evidence during post-election audits, but overwhelming evidence should be the required to override the reported electronic tally.

3. Side-Effects of Redundant Mechanisms

Redundancy is not without drawbacks. For example, function composition is itself a complex notion and properties may emerge from composing two well-understood functions that is neither easy to understand nor predictable. We consider some downsides to redundant mechanisms in this section.

3.1. The Risk of Sinking to Low[er] Assurance Development

It is important that we decide whether or not voting systems are really critical systems that demand mechanical precision and rigorous engineering. In a recent discussion regarding software independence, an esteemed colleague bemoaned the necessity of high assurance development where paper ballots were the vote of record. The intuition [and desire] of many in the voting public is that if a paper ballot is involved, the voting system will necessarily be sufficiently secure. Unfortunately, this intuition is unfounded and the trust in paper records is misplaced.

In addition to balancing the weaknesses inherent in paper-based systems noted above, we point out that electronic voting systems do much more than add one and one and one... For example, electronic voting systems:

- Provide the first count. If the first count, the one that is reported to the voting public, is incorrect, any later-reported [correct] election result will be justifiably untrusted.
- Enable rapid reporting. While we know that paper ballots, hand counted would dramatically slow the reporting process, processes that rely on electronic count would be thrown into total turmoil if they were to fail catastrophically (or less) on election day.
- Protect voter privacy. One of the wholesale threats inherent in electronic voting systems is their vulnerability to attacks that can violate voter anonymity.
- It is impossible, even if it is/were legal, to provide copies of paper ballots to every person that desired to examine them. Even if you could, it would be very difficult for anyone to conduct meaningful analysis that would have any likelihood of widespread anonymity compromise. Conversely, it is easy to provide copies of electronic ballots to anyone that asks, whenever they ask. It is also easy to analyze electronic ballots for [possibly implanted] anonymity violation hints.
- The security impact could be much more far-reaching if we were to adopt a system where electronic ballots are posted on the web.

Each of these functions is critical to establishing order in the electoral process and to restoring confidence. Moreover, this is just a small sample as electronic voting components carry out many other important functions. This leads to the conclusion that if software is used in a voting system, it must be developed for high assurance.

Ron Rivest and John Wack offer a related perspective regarding certification of composed systems in their 2006 paper describing software independence [2], where they state:

> "... one should reasonably expect the certification process should be very much more demanding and rigorous for a software dependent voting system than for a software-independent voting system."

While this notion seems reasonable to the authors, unfortunately the opposite is just as likely to be true. That is, if you combine two low assurance systems, the composition will necessarily also be a low assurance system, and its reliability is likely to be lower than either of its two components. This results from two properties of composed systems: (1) Vulnerability tends to multiply across compositions and (2) Composition tends to create vulnerability that was not present in either individual component.

Rivest and Wack's assertion implies that if a voting system is software independent, it does not need a rigorous certification process. Experience indicates that if the certification process is not rigorous, the development process will adjust to the lowest cost, and system quality will deteriorate proportionally. Thus, SI is likely to lead to continued low assurance development.

Unfortunately, low assurance engineering negates redundancy's positive impact. Comparing the merits of two systems where one is composed of two low assurance components and the other is a single high-assurance system is very complex. It is a very dangerous suggestion that because we have a backup parachute (i.e. paper), we don't need to be meticulous in packing our main parachute (high assurance software). This again leads to our strong contention that all voting system software must be developed for high assurance. It follows that any certification process should incorporate provisions that codify this notion.

3.2. Development Process Maturity

To date, many electronic voting security problems reported in public reports can be attributed to immature development processes that led to architectural, design, and implementation oriented vulnerability. While static analysis and open ended testing can incrementally improve software security properties, the greatest hope of developing consistently secure voting systems is to implement a program that assesses, tracks, and rewards vendors with mature development processes.

Process maturity management is not codified, and is only casually addressed, in the VVSG draft; likely in deference to the positive security properties that software independence offers. This is an unfortunate omission that misses a chance to make a strong systemic improvement in voting system security, reliability, and accuracy.

4. Voter Verification; Verified and Verifiable

The concept of Voter Verification continues to emerge in election terminology, sometimes in non-intuitive ways. For example, the phrase does not generally include purely electronic voting systems, where voters select races one at a time and are then presented a review screen before casting their electronic ballot, even though this is voter verification, in some sense.

On the other hand, the phrases “Voter Verified” and “Voter Verifiable” attempt to capture the essence of how voters confirm that the selections they made on one media (e.g. an electronic vote capture device) were properly transferred to a different media (e.g. the corresponding paper trail document). This voter verification during media transfer incidentally affords voters an opportunity to detect their own selection errors if any exist. More importantly, VV allows voters to detect when their selections are not properly transferred, where the transfer is usually from an electronic capture device to paper. We are interested in this notion because it is this paper record that can render an electronic voting system “software independent” as defined in the VVSG draft [1], Part 1, Section 2.7.1.

4.1. Injecting Voters Into The Security Plan

As the VV phrase evolved, ballot types once termed “voter verified” soon gave way to the less imperative “voter verifiable”, acknowledging that there was no way to force voters to verify the paper record. Voter’s tendency to not verify the paper record was troubling because in paper-trail systems, the voters themselves had, unknowingly, become a critical component in the security plan. Only the voter could detect software faults or attacks that would, for example, electronically present one selection, but record and print a different selection. If voters did not verify that the separate mediums matched, the security plan was fundamentally flawed.

The common wisdom now is that voters rarely assess the paper record, thus voter verification can contribute little to overall system security. Nonetheless, allowing voters to verify that their ballot properly transfers between media can give voters confidence and offers an opportunity to detect their own errors, both of which are good things.

The important question is: What is the impact of the acknowledgment regarding consistent lack of voter verification on the inherent security properties of software independence?

4.2. Voter Verification and Mark Sense Ballots

An interesting, and possibly confusing property of mark sense ballots is that voters cannot verify that their votes captured on paper ballots are properly transferred and captured by the standard optical scan devices in use today. This is an obvious, systematic verification gap.

The current trend is to replace reliance on voter verification with rigorous, standardized audit procedures. Though not without problems, properly conducted, random audits can detect anomalous behavior with statistically strong confidence levels.

5. Testing for Software Independence

The draft VVSG mandates that all systems submitted for certification be either software independent or be qualified through the innovation class. A glaring omission from the draft VVSG is a process description for determining if a system is SI. The draft and its supporting documents [2] indicate that VVPR and Crypto voting schemes are strongly SI, but we see no corresponding proof.

Certainly, not all voting systems that produce paper in any form are SI. More strongly, it seems that there may be systems that produce a VVPR that are not SI. It is not clear how to determine if a system meets this threshold. For example, in a contest utilizing a precinct count system in a low volume election, low percentage (5%) audit may not be able to detect malicious electoral changes that could manifest from software manipulation. More strongly, we may argue that any PCOS system that does not conduct 100% audit is software dependent.

As it stands, there no well-defined method or scientific approach to determine if a system is SI. This is a glaring omission in the VVSG draft.

6. Summary

Software independence describes a voting system architecture that offers to standardize voting systems through the certification process. This architecture ensures natural redundancy in electronic voting systems. In this paper, we offer several observations and open questions regarding security properties and prospective side effects.

We see a critical omission in the present VVSG draft relating to development processes. It is our contention that all voting systems must be engineered for high assurance accomplished through rigorous process maturity management.

7. References

1 Technical Guidelines Development Committee, “Voluntary Voting System Guidelines Recommendations to the Election Assistance Commission,” August 31, 2007

2 Ronald L. Rivest and John P. Wack, “On the notion of ‘software independence’ in voting systems,” Technical Report, DRAFT version July 26, 2006