NYSTEC

New York State Technology Enterprise Corporation

Voting System Testing Expectations Overview

For

New York State Board of Elections

Submitted to:

New York State Board of Elections
40 Steuben Place, Albany NY 12207

January 16, 2008
Version 1

Table of Contents

1. BACKGROUND

At the kickoff meetings between SysTest, NYSBOE and NYSTEC on January 8, 2008 through January 10, 2008 NYSTEC committed to producing a document that expands on the testing expectations that were discussed and that details functional and security testing expectations for the ITA's testing of voting systems. During the meetings considerable time was devoted to ensuring that SysTest understood the level of testing that was required by NYSBOE to ensure that voting systems were properly tested. In addition to testing to the requirements matrix NYSBOE is also relying on the ITA to perform sufficient testing to help ensure that NYS does not find itself in the position that other states are in when certified systems are found to contain numerous vulnerabilities that were not identified during the certification process. NYSTEC stressed that testing approaches, processes and test cases used for 2002 VVSG based testing would likely not be sufficient. NYSTEC also stated that NYS differs from other states in that they are not conducting *Open Ended Vulnerability Testing (OEVT)* but rather relying on the ITA to properly test against the 2005 VVSG and NYS requirements which include the identification of and protections against threats to voting systems. It was discussed how many of the issues with the previous ITA were related to their failure to communicate security testing expectations to test plan designers and testing engineers. This document is designed to not only expand the testing expectations discussed at the kickoff meeting but to also supply some requested example test cases to illustrate the point.

NOTE: Although this document focuses on security testing the same principles apply to all testing.

2. OVERVIEW OF TESTING EXPECTATIONS

- It is understood that SysTest will deliver to NYSBOE a Master Test Plan as well as individual Machine Specific Test Plans. Security testing will be incorporated throughout each plan and not treated separately.
- The Master Test Plan will be made available to the public and should reflect at a high level, "how" all requirements will be tested. The Master Test Plan should be detailed enough to demonstrate the ITA's full understanding of all requirements.
- Functional and security testing must be incorporated together in the overall test plans. Security and functional testing will not be separate activities.
- Test cases must strive in part, to result in effective security assessments of the voting systems. This is accomplished by designing tests that are consistent, comparable and allow for repeatable assessments of security controls that are or are not present on voting systems.

- Test cases must test not only that a security control is present, but that it is effective. To accomplish this negative testing that attempts to circumvent controls or compromise the system must be included. Negative testing should be addressed at a high level within the Master Test Plan.
- Development of test cases should utilize NIST SP800-53A as a reference. This document outlines the expected approach for the development of test plans and test cases and provides high level steps in section 3.2 that should be used as a starting point.
- The ITA must ensure that all NYS requirements (this includes all 2005 VVSG Volumes 1 & 2 requirements, NYS Law and 6209 regulations) are addressed. These requirements are in the NYS requirements matrix that was delivered to SysTest. Each requirement in the matrix must map to one or more appropriate test case(s) as needed. The mapping must be to the specific step(s) within one or more test cases that demonstrate the presence and effectiveness of a control or feature in a human repeatable manner. It should be understood that there may be "one to many" and "many to one" mapping of requirements vs. test cases and appropriate levels of detail must exist in the machine specific test cases to ensure test result repeatability.
- The master requirements matrix is to be used as a control to ensure that all requirements map to at least one test case. Additionally, the matrix can be used as a reference point for individual requirements and how they are being tested.
- The ITA must realize that in general, tests and testing methods used for 2002 VVSG EAC testing are generally considered insufficient by NYS and must undergo scrutiny if they are to be utilized. It is not sufficient to only map from the 2002 VVSG based test cases to 2005 VVSG requirements. The mapping must be done by mapping from the 2005 VVSG and NYS requirements to appropriate test cases. In many cases new test cases will need to be developed (SysTest may have already accomplished this).
- Testing against the 2005 VVSG will include testing all requirements stated as shalls as well as all requirements stated as shoulds. The shoulds, while not required of the vendor will still be tested as they will be used as tie breakers or extra credits when comparing one vendor to another.
- The ITA must focus on negative testing when designing security related tests. Tests should be designed to test the effectiveness of a control and through negative testing determine how easily the control is to circumvent. "Security is not measured by how good it works rather how bad it breaks." This point is particularly important given the fact the NYSBOE is not conducting *Open Ended Vulnerability Testing (OEVT)*. Instead, NYS is relying on thorough testing by the ITA to ensure that:

  1) Vulnerabilities previously identified (in universities, NIST, other states etc..) and on similar equipment have been addressed. (VVSG vol 1 7.4.2)

2) Vendor security features to protect against potential points of attack exist and are adequate (NYS 6209.6.F.3.n).

- The ITA developed test plans and test cases must ensure linkages between functional and security testing and the source code review where appropriate. Functional and security requirements cannot always be verified with a one-dimensional test. In many cases a directed source code review will be needed in conjunction with a functional test.
- Source code reviews should include the use of automated tools to compliment the manual review. The use of automated tools, while often generating false positives will help the evaluator focus review efforts in critical sections of code and help to identify potential problem areas that are not feasible under a strictly manual code review. It is the opinion of NYSTEC as well as our source code review partner that manual examination of source code alone is not a valid method for performing the level of testing that is required. The ITA should make extensive use of automated tools. NYSTEC is not recommending any specific tools as the ITA should utilize those with which their engineers are familiar with. Automated tools can assist the ITA with:
  1. More rapidly becoming familiar with very large and complex source code modules,
  2. Focusing manual review efforts in the appropriate code areas,
  3. Doing a context sensitive analysis over multiple blocks of code,
  4. Identification of common programming errors,
  5. Ensure compliance with good programming practices,
  6. Identification of potential buffer overflow and other memory based vulnerabilities,
  7. Identification within code of common attack points such as input/output processing, use of cryptography etc…

NOTE: There is significant discussion and recommendations in the prior test plan review documents on the NYSBOE website regarding the use of tools.

3. TEST CASE EXAMPLES

The following examples will take a NYS voting system requirement and then describe an unacceptable test case as well as an acceptable test case. This is designed to help the ITA to clearly understand the level of testing the NYSBOE is seeking. The following examples describe the level of detail NYS is seeking in the Master Test Plan. The test cases in the Master Test Plan would then be customized and detailed in the Machine

Specific Test Plan to be effective for a particular make and model of voting system or system component.

Example #1

Requirement: VVSG Volume 1 Section 2 (2.1.1.e)
Description: The system provides security provisions that are compatible with the procedures and administrative tasks involved in equipment preparation, testing and operation.

Example of unacceptable test method for VVSG Volume 1 Section 2 (2.1.1.e) :

"*Verify that the documented security provisions are compatible with the procedures and administrative tasks involved in equipment preparations, testing, and operation. Specific tasks regarding the preparation, testing, and operation shall have security provisions associated with those tasks. Verify the following:*

1. *Equipment preparation procedures references the correct security provisions*
2. *Equipment testing procedures references the correct security provisions*
3. *Equipment operation procedures references the correct security provisions*

Capture results by referencing the document and page number"

What is wrong with the above test? The above test is only referencing vendor provided system documentation and checking that the documentation correctly references the appropriate security provisions. While documentation references are certainly important there is no actual testing described here to ensure that the vendor prescribed security provisions actually function properly during use of the system and no call for related source code reviews. It should also be noted that the above test method was taken from an ITA's machine specific functional security plan! It is very apparent that this test method is completely lacking in detail and actual testing to be considered acceptable even for a master test plan test case.

Example of acceptable master test plan test method for VVSG Volume 1 Section 2 (2.1.1.e) :

1) Identify all vendor documented security provisions (should have been an output of the TDP review) that are prescribed in each phase of use of the voting system. This should be a compiled list of references to documentation where security provisions are identified. Verify that all security provisions are consonantly referenced within the vendor documentation across all phases of voting.
2) When designing the test cases, ensure that each identified security provision is incorporated into the setup for and hands on execution of all relevant test cases. Each relevant test case must contain detailed setup instruction to ensure the vendor prescribed security provisions are executed and tested.
3) Within each relevant test case, add the steps necessary to ensure that 1) the security provision is present, 2) that the security provision is actually functioning properly, 3) the security provision cannot be easily circumvented, and 4) that source code reviews are

included as needed to verify the source supports the security controls being tested in the functional testing.
4) This requirement will map to test cases that represent the multiple phases of voting as described by the vendor. During the execution of each test case where security provisions are involved each security provision will be tested and any failures, discrepancies or implementation problems will be documented.

Example #2

Requirement: NYS Election Law (7-202.1f) and related VVSG Volume 1 (2.1.8)
Description: The voting machine or system shall: be provided with a "protective counter" which records the number of times the machine or system has been operated since it was built and a "public counter" which records the number of persons who have voted on the machine at each separate election;

Example of unacceptable test method for NYS Election Law (7-2-2.1f) :

1. *Setup machine for voting*
2. *Activate voting machine for voter*
3. *Verify on voting official control panel the following:*
   - *3.1. Public counter display (indicates 0, zero)*
   - *3.2. Protective counter display (indicates a number greater than 0, zero)*
     - *3.2.1. Capture results by photograph*
4. *Verify that when the next vote is cast, both displays increment by one.*

Capture results by photograph

What is wrong with the above test? The above test case was also taken from an ITA's machine specific security test plan and would be unacceptable not only there but even within a master test plan as it is lacking in details and only tests to a very minimal level that a control is present. The test case does not evaluate the effectiveness of the control, include any corresponding source code review or negative testing. The test case also did not link back to the requirement used in the previous example.

Example of acceptable Master Test Plan test method for NYS Election Law (7-2-2.1f):

1. *Test case preparation*
   - *1.1. Reference TDP for information on how protective counter should work (Include pointer to documentation, page, etc.)*
   - *1.2. Reference corresponding source code sections that implement the protective counter (Include module name)*
   - *1.3. Using TDP and source code analyze how the protective counter and public counter was implemented*

1.3.1. Identify the provisions utilized to protect the counter from malicious access

1.3.1.1. Determine if cryptography is employed to protect the counter and if keys are maintained properly

1.3.1.2. Determine if the counter storage locations are sufficiently protected from malicious access

1.3.2. Determine if counter values can be manipulated in unintended ways

1.3.2.1. Check know vulnerabilities database for previously identified problems (i.e. some implementations have simply maintained a numeric counter in an unprotected file)

1.3.2.2. Find where in the underlying system the counter is actually stored and seek ways to compromise it (this will depend on the vendor being tested)

1.3.2.2.1. Can you boot from alternate media and access the counter as a file

1.3.2.2.2. Can an administrator simply access the file from the OS and manipulate

1.3.2.3. Use information from above analysis to develop negative tests which will be used below

2. *Setup machine for voting mode (note test would also be done in other modes)*
3. *Reference relevant source code as necessary during this test to verify security provision is being implemented as intended*
4. *Activate voting machine for voting (note test will be done in other modes as well and this would likely be part of a larger test case)*
   - *4.1. Put the machine in test mode and run test votes through it (include test case vote details)*
   - *4.2. Attempt to access the public and protected counters using analysis completed above (note it may be enough to simply identify the vulnerability and not actually exploit it)*
     - *4.2.1. If access is possible, modify the counter value (record how this was attempted)*
     - *4.2.2. Make note of the security controls that had to be bypassed*
     - *4.2.3. Determine if the system detected the access*
   - *4.3. Put machine back in voting mode*
5. *Verify on voting official control panel the following:*
   - *5.1. Public counter display (indicates 0)*
   - *5.2. Protective counter display (indicates the correct number 0 or greater from last know use)*
     - *5.2.1. Capture results by photograph and reference and save all relevant audit logs*
6. *Run 1 voter through the machine*
7. *Verify that when the next vote is cast, both displays increment by one. Capture results by photograph*
8. *Repeat step 5-6 several times and record results.*
9. *Follow procedure to close the polls.*
10. *Begin negative testing of protective counter here*

10.1. Attempt to access the protective counter from a variety of ways deemed feasible from above analysis

10.2. If access to counter is found

10.3. Modify the counter by setting it to a know value (-5, 0 etc..)

11. Proceed to step 2 and repeat through step 4, indicating if protective counter now displays the invalid value.

12. Document results of test

12.1. Where security controls were compromised, indicate how other controls could help mitigate the risk (use of hash checks, audit logs, tamper evidence seals etc...)

Example #3

Requirement: 2005 VVSG Volume 1 Section 7 (7.9.3.a)
Description: All cryptographic software in the voting system shall be approved by the U.S. Government's Cryptographic Module Validation Program, as applicable.

Example of unacceptable test method for 2005 VVSG Volume 1 Section 7 (7.9.3.a) :

1. *Check whether the voting machine has Cryptographic software.*
2. *Check which Cryptographic software is used in the machine.*
3. *Check that cryptographic software is working properly in calculating checksums, encrypting records, authentication, generating random numbers and digital signatures*
4. *Check whether the Cryptographic software is reviewed and approved by U.S. Government Cryptographic Module validation Program (CMVP).*
5. *Check that there is cryptographic voting schemes where the cryptographic algorithms used are necessarily different from any algorithms that have approved CMVP implementations*
6. *Check the feasibility of Cryptographic software whether it is required or not.*
7. *The CMVP website is http://csrc.nist.gov/cryptval.*

Capture the results by referencing the document and page number

What is wrong with the above test? The above test case was taken from a Machine Specific Test Plan and while with modifications might be sufficient for a Master Test Plan is unacceptable for a Machine Specific Test Plan because it lacks the detail to be repeatable and objective and does not focus enough on the requirement. Additionally it contains no source code review component which is necessary to determine where cryptography is used and to ensure the modules are implemented properly. Additionally, the test method should be focused on the requirement of using only FIPS approved cryptography and not on if cryptography should have been used as this would be covered in other requirements.

Example of acceptable Master Test Plan test method for 2005 VVSG Volume 1 Section 7 (7.9.3.a)

1. *Determine where the voting system (machine and EMS components) uses cryptography*
   - *1.1. Perform source code review to identify use of cryptography*
   - *1.2. Perform TDP review to identify use of cryptography*
   - *1.3. Review results of Physical Configuration Audit and Functional Configuration Audit*
   - *1.4. Perform an analysis of all files on the system to search for use of encryption*
   - *1.5. Perform an analysis of contents of removable media after use to search for use of encryption*
   - *1.6. Identify all use of cryptography on system*
2. *Verify that all cryptographic software identified in step 1 is FIPS 140-2 approved.*
   - *2.1. Verify that system vendor or partner cryptographic software is FIPS 140-2 approved by utilizing CMVP website.*
   - *2.2. Ensure that cryptographic modules are in fact only FIPS 140-2 approved modules*
     - *2.2.1. Utilize hash codes on the source modules or binary modules to verify*
     - *2.2.2. Identify any use of cryptography that is not FIPS 140-2 approved.*
   - *2.3. Verify that all cryptographic routines are implemented properly*
     - *2.3.1. Verify that the source code contains only the unmodified FIPS approved modules*
     - *2.3.2. Reference FIPS 140-2 certification and ensure that all provisions of the associated security policy have been implemented properly by the system vendor. (Note: FIPS 140-2 validation often requires use of the device according to a very specific security policy)*
       - *2.3.2.1. Verify that the vendor properly implemented key management requirements in the system and such practices are documented.*
3. *Check that cryptographic software is working properly in calculating checksums, encrypting records, authentication, generating random numbers and digital signatures*
   - *3.1. Incorporate the verification of use of cryptography in relevant test cases*

4. VOTING SYSTEM THREATS AND VULNERABILITY REFERENCES

Below are examples of typical sources that NYSTEC recommends be used to improve and validate the ITA's database of known and suspected voting system threats and vulnerabilities.

State reports and de-certifications.

Threats and vulnerabilities uncovered during current or prior testing.

http://vote.nist.gov/threats/papers.htm

http://www.elections.state.md.us/voting_system/security.html
http://www.sos.ca.gov/elections/elections_vsr.htm
http://www.sos.state.oh.us/sos/info/everest.aspx?Section=3180
http://www.elections.colorado.gov/DDefault.aspx?tid=501

http://itpolicy.princeton.edu/voting/
http://www.wijvertrouwenstemcomputersniet.nl/Nedap-en
http://www.raba.com/press/TA_Report_AccuVote.pdf
http://www.chuckherrin.com/hackthevotedemo.htm