# County of Los Angeles' VSAP Tally 2.1 Software Test Report for California

CAF-20002-*SCRTR-01*

| | |
|---|---|
| **Vendor Name** | County of Los Angeles |
| **Vendor System** | VSAP Tally 2.1 |

Prepared by:

4720 Independence St.
Wheat Ridge, CO 80033
303-422-1566
www.SLICompliance.com

***Accredited by the Election Assistance Commission (EAC) for Selected Voting System Test Methods or Services***

# Revision History

| Date | Release | Author | Revision Summary |
|---|---|---|---|
| *July 30, 2020* | 1.0 | *M. Santos* | Initial Release |

## Disclaimer

The information reported herein must not be used by the client to claim product certification, approval, or endorsement by NVLAP, NIST, or any agency of the Federal Government.

## Trademarks

- SLI is a registered trademark of SLI Compliance.
- All products and company names are used for identification purposes only and may be trademarks of their respective owners.

# TABLE OF CONTENTS

# INTRODUCTION

This report outlines the testing SLI Compliance (SLI) followed when performing Software Testing on the County of Los Angeles' Voting Solutions for All People Tally 2.1 (**VSAP Tally 2.1**) voting system against the California Voting System Standards (CVSS).

Coding languages involved in the **VSAP Tally 2.1** application are shown in Table 1.

*Table 1 –* ***County of Los Angeles VSAP Tally 2.1*** *System Languages*

| Languages | | |
|---|---|---|
| Java | JavaScript | JSX |
| C++ | CMake | CSS |
| Go | SQL | Bash |
| | Python | |

Source Code Review tools utilized by SLI included:

- ExamDiff Pro: a commercial application used to compare revised code to previously reviewed code.
- Understand: a commercial application to perform automated review of source code.
- Flawfinder: a commercial application to perform automated review of source code.

# REVIEW SPECIFICATIONS

The following are the specifications for source code testing conducted on the **VSAP Tally 2.1**.

## Software Test Review

The **VSAP Tally 2.1** includes proprietary software, the code base was tested to the applicable CVSS requirements.

Review of the code included:

- Evaluating adherence to the applicable standards in sections 5 and 7 of the CVSS.
- Evaluating adherence to other applicable coding format conventions and standards including best practices for the coding language used.
- Analyzing the program logic and branching structure.

- Evaluating whether the system is designed in a way that allows meaningful analysis, including:
  - Whether the architecture and code are amenable to an external review
  - Whether code analysis tools can be usefully applied
  - Whether the code complexity is at a level that obfuscates its logic

Security considerations reviewed against the code base included:

- Searching for exposures to commonly exploited vulnerabilities.
- Evaluating the use and correct implementation of cryptography and key management.
- Analyzing error and exception handling.
- Evaluating the likelihood of security failures being detected including:
  - Whether audit mechanisms are reliable and tamper resistant
  - Whether data that might be subject to tampering is properly validated and authenticated
- Evaluating the risk that a user can escalate his or her capabilities beyond those authorized.
- Evaluating the design and implementation to ensure that sound, generally accepted engineering practices are followed, checking to verify that code is defensively written against:
  - Bad data
  - Errors in other modules
  - Changes in environment
  - User errors
  - Other adverse conditions
- Evaluating for embedded, exploitable code (such as “Easter eggs”) that can be triggered to affect the system.
- Evaluating the code for dynamic memory access features which would permit the replacement of certificated executable code or control data or insertion of exploitable code or data.
- Evaluating the code for use of runtime scripts, instructions, or other control data that can affect the operation of security relevant functions or the integrity of the data.

# REVIEW RESULTS

## Discrepancies

Discrepancies are reported such that the California Secretary of State has a basis for evaluating the extent to which the source code meets applicable standards.

### VSAP Tally 2.1 Software Test Review

Software considerations reviewed against the source code included:

- Evaluate adherence to the applicable standards in sections 5 and 7 of the CVSS
  - The expected outcome for this review was that no issue would be found.
  - The actual outcome for this review was a determination that issues were found in the following areas:
    - ".catch()" issue, five instances (CVSS 5.2.5.a)
    - Known Language Vulnerability, one instance (CVSS 5.2.8.b.v)
    - compArray issue, one instance (CVSS 5.2.5.a)
    - Unused function issue, one instance (CVSS 5.2.7.e)
- Evaluate adherence to other applicable coding format conventions and standards including best practices for the coding language used
  - The expected outcome for this review was that no issue would be found.
  - The actual outcome for this review was a determination that no issue was found.
- Analyze the program logic and branching structure
  - The expected outcome for this review was that no issue would be found.
  - The actual outcome for this review was a determination that issues were found in the following areas:
    - .catch() issue, one instance (CVSS 5.2.5.a)
    - compArray issue, one instance (CVSS 5.2.5.a)
    - Unused function issue, one instance (CVSS 5.2.7.e)
- Evaluate whether the system is designed in a way that allows meaningful analysis, including:
  - Whether the architecture and code are amenable to an external review
  - Whether code analysis tools can be usefully applied
  - Whether the code complexity is at a level that obfuscates its logic

- The expected outcome for this review was that no issue would be found.
- The actual outcome for this review was a determination that no issue was found.

Security considerations reviewed against the code base included:

- Evaluate the use and correct implementation of cryptography and key management
  - The expected outcome for this review was that no issue would be found.
  - The actual outcome for this review was a determination that no issue was found.
- Analyze error and exception handling
  - The expected outcome for this review was that no issue would be found.
  - The actual outcome for this review was a determination that no issue was found.
- Evaluate the likelihood of security failures being detected including:
  - The expected outcome for this review was that audit mechanisms would be determined to be reliable and tamper resistant, and that any data that might be subject to tampering is properly validated and authenticated.
  - The actual outcome for this review was a determination that audit mechanisms are properly implemented to be reliable and tamper resistant, as well as that data that might be subject to tampering is properly validated and authenticated.
- Evaluate the risk that a user can escalate his or her capabilities beyond those authorized
  - The expected outcome for this review was that no issue would be found.
  - The actual outcome for this review was a determination that no issue was found.
- Evaluate the design and implementation to ensure that sound, generally accepted engineering practices are followed, checking to verify that code is defensively written against:
  - Bad data
  - Errors in other modules
  - Changes in environment
  - User errors

  - Other adverse conditions
    - The expected outcome for this review was that no issue would be found.
    - The actual outcome for this review was a determination that no issue was found.
- Evaluate for embedded, exploitable code (such as “Easter eggs”) that can be triggered to affect the system
    - The expected outcome for this review was that no issue would be found.
    - The actual outcome for this review was a determination that no issue was found.
- Evaluate the code for use of runtime scripts, instructions, or other control data that can affect the operation of security relevant functions or the integrity of the data.
    - The expected outcome for this review was that no issue would be found.
    - The actual outcome for this review was a determination that no issue was found.

Software code requirements were found to be at issue within the **VSAP Tally 2.1** source code base reviewed, as noted in this section, “VSAP Tally 2.1 Software Test Review.” As a result, discrepancies were written against the code base.

## Vulnerabilities

For any vulnerabilities discovered, SLI was tasked with identifying the particular standards applicable to each vulnerability.

To the extent possible, reported vulnerabilities include an indication of whether the exploitation of the vulnerability would require access by:

- Voter: Usually has low knowledge of the voting technology design and configuration. Some may have more advanced knowledge. May carry out attacks designed by others.
- Elections official insider: Wide range of knowledge of the voting technology design and configuration. May have unrestricted access to voting technology for long periods of time. Their designated activities include:
  - Set up and pre-election procedures;
  - Election operation;
  - Post-election processing of results; and
  - Archiving and storage operations.

- Vendor insider: Has great knowledge of voting technology design and configuration. They have unlimited access to voting technology before it is delivered to the purchaser and, thereafter, may have unrestricted access when performing warranty and maintenance service, and when providing election administration services.

SLI will not verify or demonstrate exploitability of the vulnerability but the report of the vulnerability will identify factors involved in the exploitation.

Any vulnerability theories developed by the source code review team members shall, to the extent possible, be referred to the Secretary of State staff.

## VSAP Tally 2.1 Software Code Vulnerability Review

The source code was reviewed for exposures to commonly exploited vulnerabilities, such as buffer overflows, integer overflow, and inappropriate casting or arithmetic.

- The expected outcome was that no issue would be found.
- The actual outcome was a determination that no issues were found.

The source code was reviewed for evaluation of potential vulnerabilities and related issues (code quality and standards compliance), considering that an exploitable issue in a component that is not in itself security relevant could be used to subvert more critical data. This is an issue whenever the architecture of the system does not provide strong separation of the components.

- The expected outcome for this review was that no issue would be found.
- The actual outcome for this review was a determination that an issue was found in the following area:
  - Known Language Vulnerability, one instance was noted.

  Async code does not create new threads, but simply uses the current thread. Synchronous code will block the current thread, meaning that async code will potentially receive its response late, or not at all.

  This type of vulnerability would only be exploitable by a vendor insider that would have unlimited knowledge about and access to voting technology before it is delivered to the purchaser and, thereafter, may have unrestricted access when performing warranty and maintenance service, and when providing election administration services.

The source code was reviewed for evaluation for dynamic memory access features which would permit the replacement of certificated executable code or control data or insertion of exploitable code or data.

- The expected outcome for this review was that no issue would be found.
- The actual outcome for this review was a determination that no issues were found.

## VSAP Tally 2.0 Issues

This section reviews the 10 items noted from the VSAP Tally 2.0 examination conducted in 2019, in section "5.2 Static Code Analysis & Documentation Review," in the VSAP Tally 2.0 Software Test Report, that were scheduled to be addressed in this release, VSAP Tally 2.1.

The text from each of the 10 reviewed items from the VSAP Tally 2.0 examination is *italicized* to differentiate its context with that of VSAP Tally 2.1.

***Item #15***

*Description: Third-party code provides an attack vector and must be monitored for changes and reviewed when they occur.*

*Third-party code is included which allows other coders to contribute functionality into the system. While efficient for the sake of functionality, this creates significant attack potential. Unfortunately, this potential cannot be quantified since it depends on the specific use of the third-party code in question and what functionality might be inserted.*

*The source code reviewed has 182 package.json (JavaScript Object Notation) files, the vast majority in the BMD (Ballot Marking Device) code tree. These files contain information about many third-party JavaScript tools pulled from npmjs.com, github.com, and other repositories of third-party open-source tools.*

*Third-party code is not under control of the product developer; therefore, it is the responsibility of the product developer to monitor and verify the content. The developer must also monitor publicly known vulnerabilities to be aware of flaws and fixes so they can be addressed in a timely manner.*

*The majority of third-party code referenced is maintained on github.com. This means anyone with access to those repositories can make an update to that code. A malicious contributor could conceivably update the code on GitHub knowing it will later be imported into an update of the voting system.*

*Some of the references in package.json files include SHA-1 hash values which should prevent a malicious download from being accepted (assuming this value is verified by the installation software). However, this is not always the case.*

*Some code appears to be imported from a local IP address; thus, those copies are assumed to be static and updated manually using known content and should not pose as high a risk.*

*Assessment: Use of third-party code is not in and of itself a finding, but great care must be taken to ensure malicious functionality is not introduced into code not under local control. All changes should be reviewed, no code should be included in the system automatically. The volume of third-party code and the variety of sources from which it is obtained is the finding because of the increased possibility for attack.*

*Risk may be considered acceptable provided all new code is reviewed and all imported code is verified at the time of import. Any automatic import of code from a third- party repository (e.g., GitHub) without confirmation that the content is as expected would allow for malicious injection of functionality.*

*Developer Response: All third-party code is reviewed before implementation into the system. Will continue to monitor potential threats/risks with third party software. Can provide review results of third-party code.*

*Severity: Low*

SLI Review: Third party code continues to be used. Json packages are still utilized.

Third party tools will continue to be used and monitored for potential vulnerabilities. The County and its respective developers have a process in place to monitor and if necessary mitigate.

Determination: Item is resolved.

***Item #16***

*Description: SQL database initialization seed data is entirely optional, and INSERT IGNORE can lead to unforeseen consequences.*

*The initial comment in the SQL table data import reads "This will insert static data* ***needed*** *for BMG.[1]" This indicates, or at least implies that the contents of the .sql file are required. Therefore, INSERT IGNORE is inappropriate, and should probably be removed. Another hidden feature of this syntax is that erroneous data can be imported into the database in the wrong fields because MySQL will simply massage that data to fit the field type automatically. For example, a DATETIME data piece could be converted into an INT on import because INSERT IGNORE was present.*

[1] BMG: Ballot Marking Device (BMD) manager.

*Assessment: The initial state of the BMG could be unrecoverable or badly formed data could be imported because no errors are generated. MySQL, instead of failing on a bad insert, will simply convert the data into a format that fits. In other words, INSERT IGNORE can lead to incorrect data imported into the database. Bugs generated from it could be potentially missed and therefore abused by a malicious attacker.*

*Seed data should be properly formatted to avoid insertion failures, therefore the use of INSERT IGNORE is inappropriate.*

*Developer Response: The word 'needed' in this context should be taken as 'used'.*

*Moreover, this script is used only once during deployment, and the results obtained during the tests performed are successful.*

*Severity: Low*

SLI Review: A search of the file ‘insertdata.sql’ revealed no instance of this setting.

Determination: Item is resolved.

***Item #17***

*Description: Database creation sets only_full_group_by to null, creating the possibility of inconsistent data on select.*

*In the BMG database table generation script, one line is problematic:*

*SET GLOBAL sql_mode=(SELECT REPLACE(@@sql_mode,'ONLY_FULL_GROUP_B Y',''));*

*With versions prior to MySQL 5.7.5, it would make sense to adjust/remove the group by restriction because it was considered “too strict” by most. In particular, it would enforce the use of the group by clause inappropriately in situations where it was simply not necessary. Versions of MySQL after 5.7.5 take into consideration the relative deterministic values returned from queries properly and is far less of an issue than before this version. In effect, the issues associated with ONLY_FULL_GROUP_BY have been fixed. Hence, this line is not recommended as it could lead to erroneous SQL queries that would return incorrect values, or at the very least unpredictable values due to non- deterministic queries.*

*Assessment: MySQL allows for adjusting sql_mode, such that group by restrictions aren’t maintained, which could lead to “random” results being obtained from incorrect queries.*

*This vulnerability applies to versions of MySQL prior to 5.7.5.*

*sql_mode should not be altered, so that non-deterministic queries, and therefore unpredictable values, are not returned to BMG.*

*Developer Response: The results obtained with the current BMG version code against these settings are successful. Removing this setting may cause issues.*

*Severity: Low*

SLI Review: A search of the 'BmgTableSructure.sql' file revealed no instance of this setting.
Determination: Item is resolved.

***Item #18***

*Description: Static code analysis of Go source code.*

*A scan of Go source code was performed using the GoLang Lint tool found at https://github.com/golang/lint. This produced numerous informational and warning messages. A very large number of comments concerned malformed comments around exported names or that they should be unexported. These and others may be overly cautious warnings about syntax, including things like:*

- *428 warnings of if/then/else construction*
- *204 complaints about use of += 1 and -= 1 instead of ++ and --*
- *85 warnings about range specification*

*However, other messages indicate a problem that could cause unintended behavior:*

- *12 warnings of blank imports*
- *97 warnings of returns unexported*

*Assessment: The higher potential warnings are included in an accompanying text file to this finding (i.e. same name but with a .txt extension). These should be reviewed by the development team to determine whether they could represent any issue.*

*Developer Response: The number of errors that are being reported are partially due to the repos being copied over several times. Based on the feedback there appears to be:*

- *Three copies of the Tally source code (two old and one current)*
- *Four copies of the Auth source code (two old and two current)*
- *Five copies of the Logviewer source code (three old and two current)*
- *Three copies of the Ballot Layout source code (two old and one current)*

*This increases the apparent number of errors, since the majority of the issues identified are duplicated across each copy of the repo.*

*With regards to the issues called out, all paths reviewed were inside /vendor. In Go, the vendor path is used for external dependencies (e.g. third-party libraries)*

*that were not authored by the Tally/VBL/VSAP teams. All items listed below are stock third party and occur in at least one of the following repositories:*

***Tally***

- *OLD/TDA3.local/OLD/tally-core/tally- core/vendor (appears to not be latest code)*
- *OLD/TDA3.local/tally-core/tally-core/vendor/ (appears to not be latest code)*
- *TallySource/tally-core/vendor/*

***Auth***

- *OLD/TDA3.local/auth-service/auth- service/vendor (appears to not be latest code)*
- *OLD/TDA3.local/OLD/auth-service/auth- service/vendor (appears to not be latest code)*
- *TallySource/auth-service/vendor/*
- *VBL_source_and_Keys/auth- service/vendor/Log viewer*
- *OLD/TDA1.local/logviewer-service/logviewer- service/vendor (appears to not be latest code)*
- *OLD/TDA3.local/logviewer-service/logviewer- service/vendor (appears to not be latest code)*
- *OLD/TDA3.local/OLD/logviewer- service/logviewer-service/vendor (appears to not be latest code)*
- *TallySource/logviewer-service/vendor/*
- *VBL_source_and_Keys/logviewer- service/vendor/*

***Ballot Layout***

- *OLD/TDA1.local/ballot-layout/ballot- layout/vendor/*
- *OLD/TDA1.local/ballot-layout/vendor/*
- *VBL_source_and_Keys/ballot-layout/vendor/ These entries are:*

***Warning: "exported method (or func) * returns unexported type *, which can be annoying to use":***

*These items are test code:*

*Shopify/sarama/mockresponses.go:29:59:*
*Shopify/sarama/mockresponses.go:61:60:*
*Shopify/sarama/mockresponses.go:105:64:*
*Shopify/sarama/mockresponses.go:164:62:*
*Shopify/sarama/mockresponses.go:240:61:*

*Shopify/sarama/mockresponses.go:324:71:*
*Shopify/sarama/mockresponses.go:373:70:*

*Shopify/sarama/mockresponses.go:420:67:*
*Shopify/sarama/mockresponses.go:477:62:*
*Shopify/sarama/mockresponses.go:530:66:*
*Shopify/sarama/mockresponses.go:550:67:*
*Shopify/sarama/mockresponses.go:569:67:*
*Shopify/sarama/mockresponses.go:588:71:*
*Shopify/sarama/mockresponses.go:607:68:*
*Shopify/sarama/mockresponses.go:630:70:*
*Shopify/sarama/mockresponses.go:656:67:*
*Shopify/sarama/mockresponses.go:677:65:*
*Shopify/sarama/mockresponses.go:695:63:*
*Shopify/sarama/mockresponses.go:717:65:*
*stretchr/testify/mock/mock.go:620:32 testify/mock/mock.go:532:32*

*Production code written to allow for testing:*

*gocql/gocql/host_source.go:286:30: gocql/gocql/host_source.go:299:28*
*hashicorp/go-sockaddr/ifaddrs.go:46:49 hashicorp/go-sockaddr/route_info_bsd.go:17:22 hashicorp/go-sockaddr/sockaddrs.go:32:45*
*modern-go/reflect2/reflect2.go:136:27*
*k8s.io/apimachinery/pkg/util/strategicpatch/typ es.go:48:50*
*k8s.io/apimachinery/pkg/util/strategicpatch/typ es.go:111:51*

*Although the linter is correct that this can be annoying, this is done intentionally in test code where a mock object is returned that implements the same interface as the real object to allow for better control and injection of test harnesses into unit test code.*

*In production code this pattern allows unit tests to simulate the state the code under test is running in to better check code behavior.*

***Warning: "a blank import should only be in a main or test package, or have a comment justifying it":***

*This error only occurs in support packages officially published by the Go team (although it occurs in several copies of the tally-core repo that were scanned:*

*golang.org/x/crypto/openpgp/read.go:10:2*
*golang.org/x/crypto/openpgp/packet/public_ke y.go:15:2*
*golang.org/x/crypto/ssh/common.go:15:2*

*The same warning**: "a blank import should be only in a main or test package, or have a comment justifying it"** does occur once in a library that the ballot layout team has modified. This code ("bitbucket.org/vsap/pdf/image_obj.go:7:2") occurs three times in the scan results as the results seem to include three copies of the VBL repo. Although this is a library that we had to modify, this file remains unchanged. When updating the library, it was deemed safer to leave imports that*

*we were not impacting alone rather than trying to change things that could have been done stylistically better.*

*Severity: Low*

> SLI Review: A scan of Go source code was performed using the GoLang Lint tool found at https://github.com/golang/lint. This produced numerous informational and warning messages, as per the VSAP Tally 2.0 Findings.
>
> This finding was/is informational, as the tool used scans for “preferences” in the code. Code, particularly that which is custom developed, is at the discretion of the customer. Functionally, this finding has no impact on the system.
>
> Determination: Item is resolved.

***Item #19***

*Description: Static code analysis of JavaScript source code.*

*A scan of JavaScript code was performed using the JavaScript Lint tool found at javascriptlint.com. This produced numerous informational and warning messages. Many may be overly cautious warnings about syntax, including things like:*

- *1,973 warnings block statements containing block statements should use curly braces to resolve ambiguity*
- *1,464 warnings that comparisons against null, 0, true, false, or an empty string allow implicit type conversion so === or !== should be used*
- *858 warnings that functions do not always return a value*
- *118 warnings of variable redeclarations*
- *20 warnings where using parseInt is missing a radix parameter (but default behavior is defined)*
- *12 warnings about missing default case statements in a switch statement*

*However, other messages indicate a problem that could cause unintended behavior:*

- *207 warnings that an else statement could be matched with one of multiple if statements*
- *48 warnings of useless comparisons using identical expressions*
- *21 warnings of unknown order of operations for successive plus or minus signs (x+++y or x---y)*

*Assessment: The higher potential warnings are included in an accompanying text file to this finding (i.e. same name but with a .txt extension). These should be reviewed by the development team to determine whether they could represent any issue.*

*Developer Response: In this item, like 18, it appears that several repositories are mixed together. We are ignoring the "OLD/BMD_Code/", as that is not our area to respond. We are also ignoring:*

- *OLD/TDA1.local/ballot-layout/**
- *OLD/TDA1.local/logviewer-service/**
- *OLD/TDA1.local/vbl_deployment/**
- *OLD/TDA3.local/OLD/auth-service/**
- *OLD/TDA3.local/OLD/logviewer-service/**
- *OLD/TDA3.local/OLD/tally-core/**
- *OLD/TDA3.local/auth-service/**
- *OLD/TDA3.local/logviewer-service/**
- *OLD/TDA3.local/tally-core/**

*These paths/repos seem to have been superseded by:*

- *TallySource/auth-service/**
- *TallySource/logviewer-service/logviewer/**
- *TallySource/tally-core/**
- *VBL_source_and_Keys/auth-service/**
- *VBL_source_and_Keys/ballot-layout/**
- *VBL_source_and_Keys/logviewer- service/**

*Even here there is a significant amount of duplication, but it brings the total number of findings down to 91. Further review shows that these are actually only 13 distinct issues. Twelve are in Jquery in the file "jquery-3.2.1.min.js"*

- *2:lint warning: useless comparison; comparing identical expressions*
- *2:lint warning: the else statement could be matched with one of multiple if statements (use curly braces to indicate intent)*
- *2:lint warning: the else statement could be matched with one of multiple if statements (use curly braces to indicate intent)*
- *3:lint warning: the else statement could be matched with one of multiple if statements (use curly braces to indicate intent)*
- *3:lint warning: the else statement could be matched with one of multiple if statements (use curly braces to indicate intent)*
- *3:lint warning: the else statement could be matched with one of multiple if statements (use curly braces to indicate intent)*
- *3:lint warning: the else statement could be matched with one of multiple if statements (use curly braces to indicate intent)*
- *3:lint warning: the else statement could be matched with one of multiple if statements (use curly braces to indicate intent)*
- *3:lint warning: useless comparison; comparing identical expressions*

- *4:lint warning: the else statement could be matched with one of multiple if statements (use curly braces to indicate intent)*
- *4:lint warning: the else statement could be matched with one of multiple if statements (use curly braces to indicate intent)*
- *4:lint warning: the else statement could be matched with one of multiple if statements (use curly braces to indicate intent)*
- *4:lint warning: unknown order of operations for successive plus (e.g. x+++y) or minus (e.g. x---y) signs*

*Jquery is a major project. While we have not analyzed these findings code use cases, there seem to be no CVEs related to them. Additionally, this is checking minified code - meaning that it has been post processed to make it as small as possible. It appears that most of these warnings are stylistic to avoid confusion, as such, while valid in code that would be read by humans, are likely not relevant to minified source code as the computer will not treat them as ambiguous or unclear.*

*There was also one identified issue in bootstrap- table.min.js (although it was identified multiple times) that appears to be the same case as the JQuery issues.*

*Severity: Low*

SLI Review: A scan of JavaScript source code was performed using the JavaScript Lint tool found at javascriptlint.com. This produced numerous informational and warning messages, as per the VSAP Tally 2.0 Findings.

This finding was/is informational, as the tool used scans for “preferences” in the code. Code, particularly that which is custom developed, is at the discretion of the customer. Functionally, this finding has no impact on the system.

Determination: Item is resolved.

***Item #20***

*Description: Public vulnerability search. See sections 4.2, Published Vulnerabilities, and 5.1, Public Vulnerability Search, for complete results.*

*Assessment: The system is air-gapped; that is, not connected to the internet or connected to any other system that is connected to the internet.*

*Air gap systems include*

- *Ballot Marking Device Manager (BMG)*
- *Ballot Marking Device (BMD)*
- *VSAP Ballot Layout (VBL)*
- *Tally*

*The following security products are used to facilitate the air-gapped environment:*

- *Carbon Black Protection: Provides application control to lock down critical systems in order to prevent unwanted software changes and malicious attacks.*
- *CylancePROTECT: Threat prevention solution (anti-virus) which utilizes machine-learning, allowing the software to function in isolation from the internet or cloud connection.*
- *HP Aruba ClearPass: Tracks machine (MAC) addresses of all network cards on the network and can remove unauthorized addresses.*
- *Net Fort LANGuardian: Tracks movement of all software, users, and actions on the network.*
- *Snare System Information and Event Management (SIEM): Records all computer system and network activities, which are available for review in the event of an attack or issue.*
- *Thycotic Secret Server: Manages all administrative privileged network accounts and limits users to standard access, limiting opportunities for software changes.*

*Note: Unused hardware ports (i.e. USB ports) are protected by port locks and/or tamper evident seals with signaling residue to reveal modification and/or removal. The serialized tamper evident seals are manually logged with an operator signature, seal number, location, date, and time. This is to prevent removal of authorized connections when the port is in use and to prevent the insertion of unauthorized connections when the port is not in use. This prevents any infected USB flash drive from crossing any air gap.*

*Developer Response: Smartmatic staff has investigated this list of potential software security vulnerabilities. We find that most of these relate to Internet connected systems. Some could be exploited by trusted insiders, even without the system being inadvertently or maliciously attached to the Internet. We note that many are not easy to exploit or would not give an attacker meaningful access or capabilities that would allow undetectable manipulation or results or denial of service. A malicious trusted insider would likely attempt other avenues by which to subvert the voting system.*

*We would like to point out that the entire certification candidate VSAP voting system remains under contracted Warranty for two years, and optional (Los Angeles County carries the option) maintenance beyond that timeframe.*

*These listed software vulnerabilities as well as others that might be found in the future by researchers would be dangerous if the product is off support, meaning that no one is available to assess the vulnerability and remediate it if deemed necessary. CVSS speaks to the possibility that new, unforeseen vulnerabilities in voting systems may emerge during the system lifecycle. In several places (9.6.d and 9.6.3.g as two examples) CVSS requires planning to respond to new threats.*

*Los Angeles County has fulfilled the letter and spirit of these clauses by ensuring that their System Integrator remains responsible for system maintenance.*

*At this late time in the Certification campaign, we do not see the ability to remediate the listed software vulnerabilities assuming any could be exploited and would serve as a valuable target. Where deemed necessary by Los Angeles County, the system owner and operator, or the Secretary of State vulnerabilities will be remediated under the established system warranty contract clauses.*

*Severity: Low*

SLI Review: Of the products listed, only the following are still in use:

- Carbon Black Protection
- HP Aruba ClearPass (BMG only)
- Net Fort LANGuardian(BMG only)
- Snare System Information and Event Management

Because the system is air gapped and strict security protocols, including access are followed, this is not an issue. Additionally, this finding is better suited in the Security/Telecom area.

Determination: Item is resolved.

***Item #21***

*Description: The CA certificate and key are stored in tmp and set to 777 file permissions. Programmatic copy of the CA cert and key to the cluster machines makes sense as its going to be necessary for later steps in the process, but in this case the file is set to 777 permissions, which means that all users have all permissions on these files.*

*While it's possible that Kubernetes requires liberal file permissions for its CA cert and key, the go script does not delete the source files. Thus, the CA key could be stolen and used to falsify certificates.*

*Assessment: Programmatic setting of permissions to highly open configuration, and source files are not deleted after being copied to destinations on cluster machines.*

*Leaving a copy of the CA key in the temp folder of a multi-user operating system is an incorrect configuration of a CA or PKI infrastructure. Industry standard processes dictate that the root CA is created and stored on an air- gapped system, and intermediate CA's used to further certificate generation on destination machines. If this is the root CA in particular, then this is an inappropriate use case. If nothing else, the environment should be cleaned to prevent the CA from falling into the wrong hands.*

*Developer Response: In practice, this isn't a significant risk as, although the operating system is multi-user, the machine cluster is single tenant running only*

*the Tally (or VBL) system and only administrators on the Tally system should be authorized on the environment.*

*Mitigation*

- *The documentation will be updated to instruct the installer user to delete all data from temp once the install is finished.*
- *A procedure has been added to restrict file system permissions on these files post-install.*

*Severity: Low*

SLI Review: The file "VBL v1.1.3 Build Guide v1.2(06.24.2020).pdf" contains section 5.4, which details how to remove the contents of the 'temp' folder after successful installation.

Determination: Item is resolved.

***Item #22***

*Description: Point of origin is not taken into consideration with authentication entries.*

*Authentication with MySQL can also factor in point of origin which should be used to prevent unauthorized attempts to login. The SQL template allows authentication to the MySQL host from any other host for all users defined.*

*The template file uses the standard syntax of '%' to signify the any host wild card for users being inserted into the database.*

*Assessment: This configuration could allow someone to systematically try different authentication combinations until a valid one is found, leading to invalid voting data.*

*Unless it's crucial that all users can login from all hosts, then the default template is too liberal in its use and definitions of who can login from where.*

*Developer Response: The user must be able to log in from a docker container on one of several (currently about nine) kubernetes cluster machines. Moving forward, we can look at ways to limit this host list, but at present this would appear to require making some significant assumptions about the details of the production environment (such as IP addresses) that pose a challenge.*

*Moving forward we will look for better options to lock this down. We may be able to implement a manual procedure for more specific grants if this is deemed a high priority issue.*

*Severity: Low*

SLI Review: Evaluation of the template file did not reveal usage of the standard syntax of '%' to signify the any host wild card for users being inserted into the database.

Determination: Item is resolved.

***Item #24***

*Description: Python 2 reaches end of life at the end of this year and will not be supported after 2019-12-31. Any future security vulnerabilities found in Python 2 will not be fixed.*

*The document VSAP-TDP- 012_Approved_Parts_List.pdf indicates both Python 2 and Python 3 are used. If Python 2 is a dependency, the product uses a language that is no longer supported after January 1, 2020.*

*Assessment: While this does not represent an actual vulnerability, it has the potential to cause one in the future. Python 2 will not be supported or updated starting in January 2020.*

*If any security vulnerabilities are found after that date, not only could they put the voting system at risk, they would most likely not be fixed. Developers should already be in the process of migrating code to Python 3. Please see https://www.python.org/do c/sunset-python-2/ .*

*Developer Response: We reviewed open CVEs for Python 2.7 (the version used in the BMD) and found none that are scored in the 8, 9, and 10 range. We also note that the VSAP BMD remains under contracted warranty for two years, and optional maintenance beyond that timeframe. Python 2 vulnerabilities that might be found by researchers in the future would be dangerous if the product is off support, meaning that no one is available to assess the vulnerability and remediate it if deemed necessary. CVSS speaks to the possibility that new, unforeseen vulnerabilities in COTS products may emerge during the system lifecycle. In several places (9.6.d and 9.6.3.g as two examples) CVSS requires planning to respond to new threats.*

*At this late time in the certification campaign, we do not see the ability to move to Python 3 in the BMD software; however, we plan to fulfill the letter and spirit of CVSS and will monitor for new vulnerabilities in Python 2 during the warranty phase of VSAP lifecycle. Where deemed necessary by Los Angeles County, the system owner and operator, or the Secretary of State new Python 2 vulnerabilities will be remediated under the warranty contract clauses.*

*Severity: Low*

SLI Review: No implementation of Python 2 was found, only Python 3.

Determination: Item is resolved.

***Item #25***

*Description: Calico container securityContext set to privileged = true.*

*securityContext: true is set for the container Calico, which controls network functions.*

*Assessment: The potential problem with this configuration is simply that the container is running effectively as root. An attacker could use this to reboot the system, delete files, modify passwords, etc.*

*However, there is a bug report filed at the following URL, which is attempting to deal with this issue related to Calico: https://github.com/projectcalico/calico/issues/2000*

*That said, it should be mentioned as a future improvement for the voting system, as this level of access to a machine via container is unnecessary and dangerous.*

*Developer Response: We agree that this is not an emergent finding but a future system version could see this remediated.*

*Severity: Low*

SLI Review: No update to this setting has been implemented.

Determination: Item is continuing to be monitored, and will be addressed in a future version pending action from Calico, a third party.

# Final Report

Findings were identified for the **VSAP Tally 2.1** code base, as identified in the Review Results/ VSAP Tally 2.1 Software Test Review section above.

Potential vulnerabilities were identified within the **VSAP Tally 2.1** code base, as identified in the Review Results/ VSAP Tally 2.1 Software Code Vulnerability Review section above.

As directed by the California Secretary of State, this software testing report does not include any recommendation as to whether or not the system should be approved.

# End of VSAP Tally 2.1 Software Test Report