4/9/2021

Antrim County, MI

Election Management System
Application Security Analysis

Cyber Ninjas
Phone: (941) 3-NINJAS
Fax: (941) 364-6527
www.CyberNinjas.com
5077 Fruitville Rd #109-421, Sarasota, FL 34232

1 Revision History

Date	Revision	Notes
01/10/2021	DRAFT	Initial Draft Created
01/11/2021	DRAFT	Revision 1.0 Completed
03/13/2021	DRAFT	Additional Findings Added
04/07/2021	DRAFT	Additional Findings
04/08/2021	DRAFT	Revision 2.0 Completed

2 Table of Contents

3 EXECUTIVE SUMMARY

The Antrim County Election Management System (EMS) environment is setup in a manner where it would potentially be possible for an individual to alter the results of the election without leaving much of a digital trail.

- Users of the computer have enough access rights and the needed tools installed to directly modify election results in the database. Official results are generated from this database.
- The master encryption key utilized to encrypt election results is stored in plain text in the database, and its value exists both at the county and with Election Source. If Election Source was hacked, or this value otherwise got into a malicious actor's hand; it would be possible to create malicious tabulator configurations or alter the result files from tabulators. Either of these could be used to change the results of an election.
- Log levels are such on the system that it would be possible to delete files, delete logs, or the similar; and it would be difficult to have the necessary details available to investigate the incident.
- Application and computer system accounts are generic and shared among multiple individuals making it near impossible to determine who performed an action even if proper logging was in place.
- Hard-coded credentials, failure to use cryptography properly, and other well-known bad practices are utilized throughout the software suggesting that exploitation of the software is very possible. These types of problems are documented to be reoccurring with this EMS going back over 10 years.
- Ballot images are missing from the Compact Flash data, making it difficult to audit how the software interpreted any given ballot.

These types of findings and departures from best practices utilized across multiple industries for over 10 years is inexplicable for a system that is both highly sensitive, and a likely target for nation state activity.

It is highly recommended that all components of the EMS software immediately go through a full code-review audit to determine the extent of the problems encountered and how easily other areas of the application may be exploitable. In addition, it is recommended that the following items be reviewed to have a better understanding of the full impact of some of these findings:

- Election Source should be required to provide a list of all personnel that have access to the Election Definition databases utilized for Antrim County, as well as provide documentation on any controls that are in place to detect and prevent a breach or modification of election data.
    - Should the controls be determined to be insufficient to detect a nation state level attack, at a bare minimum; all Michigan election projects, and compact flash cards should be forensically imaged and reviewed to determine if any alterations of the data or systems took place.
- Documentation should be requested on the reasoning for installing Microsoft SQL Management Tools onto the EMS Server and who performed this action. This software is not on the EAC's approved list for certified systems, and a legitimate purpose for its installation is not apparent. Yet this software greatly facilitates the changing of database values.
- Copies of the tabulator tape result output for all precincts should be provided, in addition to chain-of-custody documentation showing that these files have been properly cared for and have not been altered. These numbers should then be compared against the numbers read directly from the compact flash cards.
- File definitions should be provided by Dominion for the various results and configuration files held on the compact flash cards, so that the decrypted files can easily be read and confirmed to match the EMS Server and therefore no alteration took place.

4 SCOPE

Cyber Ninjas was engaged to evaluate the security of the Election Management System (EMS) utilized in Antrim County, MI in order to determine if cyber security related flaws, abuse of functionality, misconfiguration or purposely malicious actions or code could account for the voting irregularities demonstrated in the county during the November 2020 General Election.

A forensic image of the Antrim County Election Management System (EMS) gathered on December 6th, 2020 was converted to a bootable virtual machine. This machine was then utilized to allow the EMS to be utilized in a "live" environment to examine logs, configurations, and functionality of the applications. All analysis was performed within this virtual environment.

5 BACKGROUND

The following section outlines background details and definitions useful in understanding the overall Election Management System (EMS) architecture and structure, as well as definitions that are utilized throughout the report. Architecture details came from publicly available documentation, as well as reviewing the deployment within Antrim County.

5.1 Architecture

The architecture of the Election Management System (EMS) in Antrim County consisted of one or more ImageCast Precinct (ICP) tabulators and an ImageCast X (ICX) Ballot Marking Device (BMD) at every precinct, as well as the EMS Server that was centrally located at the county. While the ICP devices support a number of different ways to remotely report results, Antrim County stated that they aggregated the results by collecting the compact flash cards and manually importing the results off of these compact flash cards.

5.1.1 ELECTION MANAGEMENT SYSTEM SERVER (EMS SERVER)

The EMS Server is the primary device utilized in Antrim County in order to run an election and serves as the central aggregator for all election results within the county. While the EMS Election Event Designer software can be utilized to build an entire election from scratch, documentation provided indicates that the initial election definition was created by Election Source and exported as a package for Antrim County to then import into their system. This configuration was then utilized by the county in order to build the compact flash cards utilized to configure the ImageCast Precinct (ICP) devices, and after the polls were closed these same compact flash cards were brought back to the EMS in order to attempt to import and publish the results. The EMS software also supports the manual entry of result files.

The EMS Server machine in smaller counties can at times also be utilized as the digital adjudication machine, but this software was not installed on the EMS Server image that was reviewed. Examining the Windows Event Logs shows that the DVS Adjudication Services software had been installed on April 10, 2019; but had later been removed on September 3rd, 2019. This explains the DVS Adjudication logs from 2019 referenced from the ASOG report, and also explains why there were not any adjudication logs for 2020. This is consistent with what the county has reported that all adjudication for 2020 was done manually.

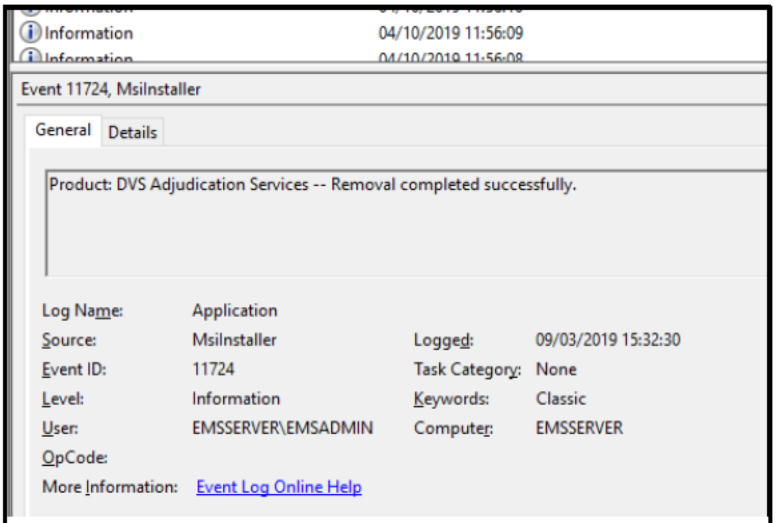


Figure 1 - DVS Adjudication Services software uninstalled 9/03/2019.

5.1.2 ImageCast X (ICX) - Ballot Marking Device (BMD)

ImageCast X (ICX) Ballot Marking Devices (BMD) are primarily utilized in Antrim County in order to support accessibility voting. With these devices an individual can vote via a touch screen or a number of specialized input devices, and this in turn prints out a ballot with a QR code which can then be fed into the ImageCast Precinct (ICP) tabulators where the votes are tabulated. These devices are configured utilizing USB drives created from the EMS Server.

5.1.3 ImageCast Precinct (ICP) – Precinct Tabulator

ImageCast Precinct (ICP) devices are designed to handle the tabulation of ballots at a precinct level. These devices support both a standard hand-filled out bubble ballot, as well as the QR coded ballots created by the ICP device. As votes are cast the results are written simultaneously to two compact flash cards for redundancy. One of these compact flash cards also holds all the configuration for the ICP device. This configuration tells the ICP how to read and understand the various ballot types for the election. These compact flash cards are built on the EMS Server.

5.2 Definitions

The following section references various definitions to help clarify their meaning throughout the report, and to clear up some common misconceptions related to the systems reviewed.

5.2.1 Log Files

The term log files will be utilized for any place where an application or the operating system writes information about what happened as an audit trail, or to aid in debugging. This includes, but is not limited to, specialized and general Windows Event logs, the slog.txt files from the ICP tabulators, as well as the UserLog database table found within every election database.

5.2.2 Source Code

Source Code is the text which is written by a programmer which can be compiled into a program. The compiled program is then deployed onto a computer to perform the desired action. There was no source-code encountered on any of the compact flash drives, or on any of the forensic images captured. Only compiled programs were deployed on these systems.

6 Findings

The following sections outline the findings discovered over the course of the analysis. This included significant deviations from application security best practices, configurations that could allow the integrity of the system to be compromised, and suspicious actual usage of the EMS. This information is broken out by topic with sections that cover “Authentication & Authorization”, “Audit Logging & Tracking”, “Cryptography & Secrete Storage”, “Credential Management”, and “Tabulation Irregularities”.

6.1 Authentication & Authorization

The application did not follow best practices related with assigning least privilege to the users required to do the job or implement proper account management. This represents a large risk to the integrity of the election data. With the current setup it would be simple for a malicious admin to modify the vote in a manner where it would be difficult to determine who did it.

6.1.1 System Administrator utilized for all Access

The only user that has been utilized to login to the EMS Server machine for the entire history available in the Security Event Logs is the use EMSADMIN. This means that the EMSADMIN user is being utilized for normal, everyday use of the various applications associated with voting. This is a huge security risk and could easily be utilized to compromise or change the entire vote.

The EMSADMIN user is a full administrator on the machine in addition to being a full administrator on the Microsoft SQL Database utilized to store all election data. Furthermore, the EMSSERVER has the appropriate tools installed to make it simple to manually update any value in the database. This means that regardless of what level of access a user has within the EMS application they’d be able to change anything they wanted because they’re accessing the computer as an admin.

This could be utilized to:

- Change vote totals within the database affecting final results.
- Add, Edit, or Delete any user in the application, including changing passwords.
- Delete any sort of logs or audit trail that may exist on the computer, or in the database.

6.1.2 System and Application Shared Accounts

The application utilizes generic usernames and passwords rather than creating usernames for the individuals that will be utilizing the application. This likely means that more than one user has the credentials to the same account in order to perform various election related operations in the application. This defies best practice and makes it impossible for you to know who it was that performed a given operation within the application since multiple people have access to the same username. Best practices dictate that each user should always have his or her own username and password to the application. This increases accountability and helps avoid situations where credentials might be leaked.

6.2 Audit Logging & Tracking

The EMS server configuration fails to implement audit logs and controls that would be typical of a high-risk application. In many cases this would prevent the audit logs from existing that would be required to look into or detect a security incident.

6.2.1 No Ballot Images

None of the Compact Flash drives appeared to hold ballot images, and no ballot images had otherwise been imported into the EMS. Ballot images are a critical artifact and are essential for any type of system audit to determine how an electronic voting machine interpreted results and where it might be malfunctioning. Vendor training clearly state that ballot images should be imported into the EMS immediately following the election, but this was never done, and the images don't even seem to be present. Without ballot images its near impossible to match up and see the origin of where errors might be happening.

It is unclear how write-in candidates could have been properly handled without ballot images available for review.

6.2.2 Insufficient Audit Logs

Audit logs should be configured in a manner that all sensitive operations are logged, that the logs include all details necessary to investigate suspicious activity, and that the logs are difficult to tamper with. This was not the case with the Windows Event logs, nor the EMS Application logs.

The Windows Operating System is configured with the standard log level which does not log the access of sensitive files, the deletion of files, or other sensitive actions. This is atypical for a machine that is as sensitive as serving as the central aggregator of all the votes in the county.

The EMS logs found in the UserLog table are also completely deleted any time an election package is loaded within EED. Loading an election package in this way is a standard way that organizations such as Election Source provide the election event. However, this would mean it would be possible for someone to set a malicious device configuration, build the compact flash cards; and then reset the database and put things back to normal. This process would destroy all evidence of the change. Furthermore, the user, EMSADMIN, which is the main account utilized on the machine; has full access to edit the database and delete any log entries. Best practices dictate that an account utilized for normal use of a system not have access to edit or remove logs.

6.2.3 Manual Entries Do Not Require A Comment

The Result Tally and Reporting application can be utilized to insert manual vote count totals rather than automatically importing those results from the tabulator. These manual entries appear to be a way to override and replace the existing vote totals rather than allowing an interface where the numbers that are pulled in from a tabulator can be adjusted with some sort of audit trail. This interface does not log the username submitting the details, require a comment explaining the changes, or even display a timestamp so it was clear when the manual count was done.

These type of entries and comments are standard for any inventory or financial services application. It seems the sensitivity of an election system would be higher than that of these systems.

6.3 Cryptography & Secret Storage

The application did not appear to follow best practices for credential storage. The full extent of the problem cannot be fully determined without a review of the actual source code. However, simply by working with the files on the file system and looking in the database it was possible to find various sensitive details that are not properly stored. This included everything from the master encryption key to hard coded credentials.

6.3.1 Plaintext Cryptographic Keys

The master cryptographic key utilized to encrypt all voting results and configuration from the tabulators is stored in plain text in a table within the database for this election. With this key and knowledge about the file formats utilized; it would be possible to alter election results prior to those result files being loaded into the EMS Server, or to alter configurations for the tabulators to make them behave in a certain way. Furthermore, since Election Source originally built the election package utilized by the county and is the originator of the database; any employee at Election Source who had access to the county's database file, or any nation state that compromised one of their computers; would have the encryption key needed to adjust files on the compact flash cards.

Best practices would dictate that any encryption key utilized for election files would only exist on the County's EMS Server and stored in a hardware Trusted Platform Module (TPM). Since the compact flash cards for the tabulators are always built locally, there is no reason for this encryption key to exist anywhere except for the location where the cards are built. Failing to do so significantly reduces the overall security of the election.

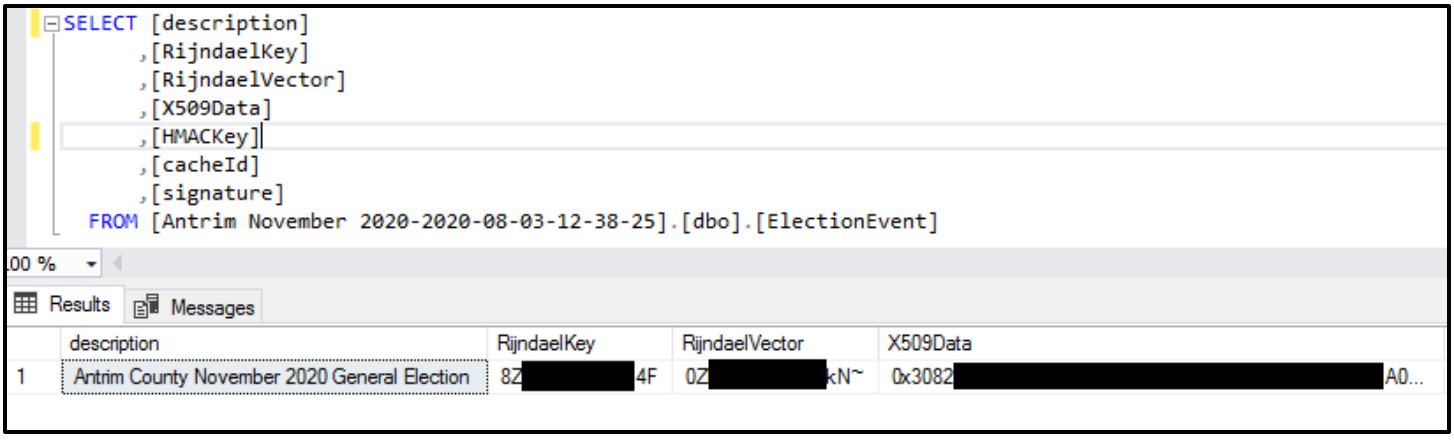


Figure 2 - Cryptographic keys are in plain-text in the databases. The values are redacted for security purposes.

6.3.2 Hard coded Credentials

Components of the EMS have hard-coded credentials compiled within the application itself. This is considered an extremely bad practice and is not something that should ever be done. Not only can credentials be exposed when they're hard coded in the application, but the fact that they're compiled in the application means that every single customer of this version of the EMS would utilize the same credentials. As a result, learning the credentials would allow you to attack them all.

Hard coding of credentials into this application appears to go back to at least 2010, based on the following report:

https://www.eac.gov/sites/default/files/voting_system/files/Deficiency%20Report.pdf

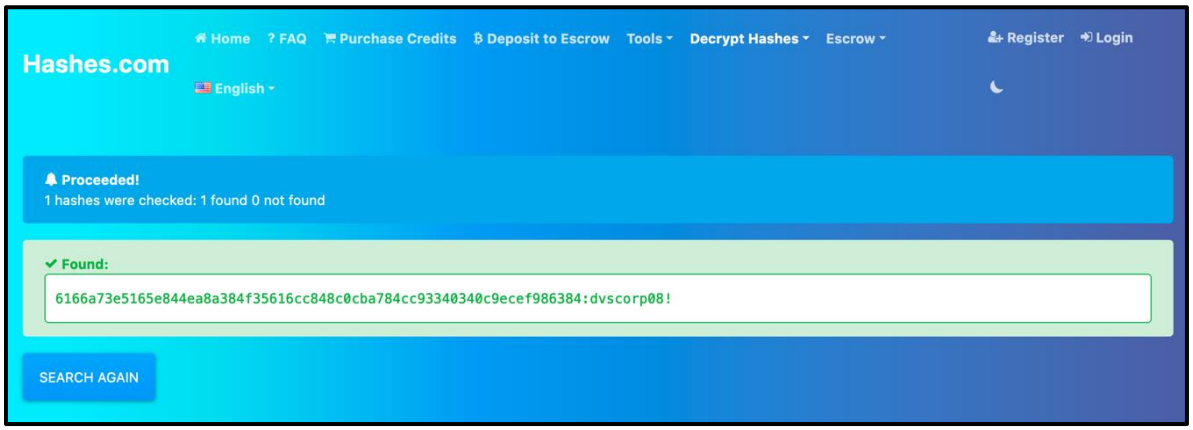
NOTE: These were detected by utilizing grep to search the binaries for the string "password".

File Name(s)	Value
/Election Data Translator/DVS.DemocracySuite.DatabaseService.dll /Election Event Designer/DVS.DemocracySuite.DatabaseService.dll /Election Event Designer/DVS.DemocracySuite.DatabaseService.dll	username="Techadvisor" password="YWanPlFl6ETqijhPNWFsyEjAy6eEzJM0A0DJ7O+YY4Q="
/Election Data Translator/DVS.DemocracySuite.DatabaseService.dll /Election Event Designer/DVS.DemocracySuite.DatabaseService.dll /Election Event Designer/DVS.DemocracySuite.DatabaseService.dll	username="MRO01" password="YWanPlFl6ETqijhPNWFsyEjAy6eEzJM0A0DJ7O+YY4Q="
/Election Data Translator/DVS.DemocracySuite.DatabaseService.dll /Election Event Designer/DVS.DemocracySuite.DatabaseService.dll /Election Event Designer/DVS.DemocracySuite.DatabaseService.dll	username="ROAdmin" password="YWanPlFl6ETqijhPNWFsyEjAy6eEzJM0A0DJ7O+YY4Q="
/Election Data Translator/DVS.DemocracySuite.DatabaseService.dll /Election Event Designer/DVS.DemocracySuite.DatabaseService.dll /Election Event Designer/DVS.DemocracySuite.DatabaseService.dll	username="Admin" password="YWanPlFl6ETqijhPNWFsyEjAy6eEzJM0A0DJ7O+YY4Q="

/Election Data Translator/DVS.DemocracySuite.DatabaseService.dll /Election Event Designer/DVS.DemocracySuite.DatabaseService.dll /Election Event Designer/DVS.DemocracySuite.DatabaseService.dll	username="SAdmin" password="YWanPlFl6ETqijhPNWFsyEjAy6eEzJM0A0DJ7O+YY4Q="
/Election Data Translator/DVS.DemocracySuite.DatabaseService.dll /Election Event Designer/DVS.DemocracySuite.DatabaseService.dll /Election Event Designer/DVS.DemocracySuite.DatabaseService.dll	username="Admin" password="oCFR3h+mPKykHkkE41o5cvyCSwY="
/Election Data Translator/DVS.DemocracySuite.DatabaseService.dll /Election Event Designer/DVS.DemocracySuite.DatabaseService.dll /Election Event Designer/DVS.DemocracySuite.DatabaseService.dll	username="Admin" password="oCFR3h+mPKykHkkE41o5cvyCSwY="

6.3.3 Passwords Stored As An Unsalted Hash

Credentials to the EMS applications are stored within the Microsoft SQL Database utilizing the hashing algorithm SHA256. This is better than storing the credentials in the database as plain text, but industry best practices dictate that these passwords should also have a cryptographically random string tacked onto the front of them before being hashed. This is referred to as “salt”; and it prevents several common attacks that might allow an attacker to figure out the credentials. Because salt was not used, we were actually able to take the hash out of the database, 6166A73E5165E844EA8A384F35616CC848C0CBA784CC93340340C9ECEF986384, and run it through a database of pre-computed hashes at https://hashes.com/. This let us figure out that its value was, “dvscorp08!”.



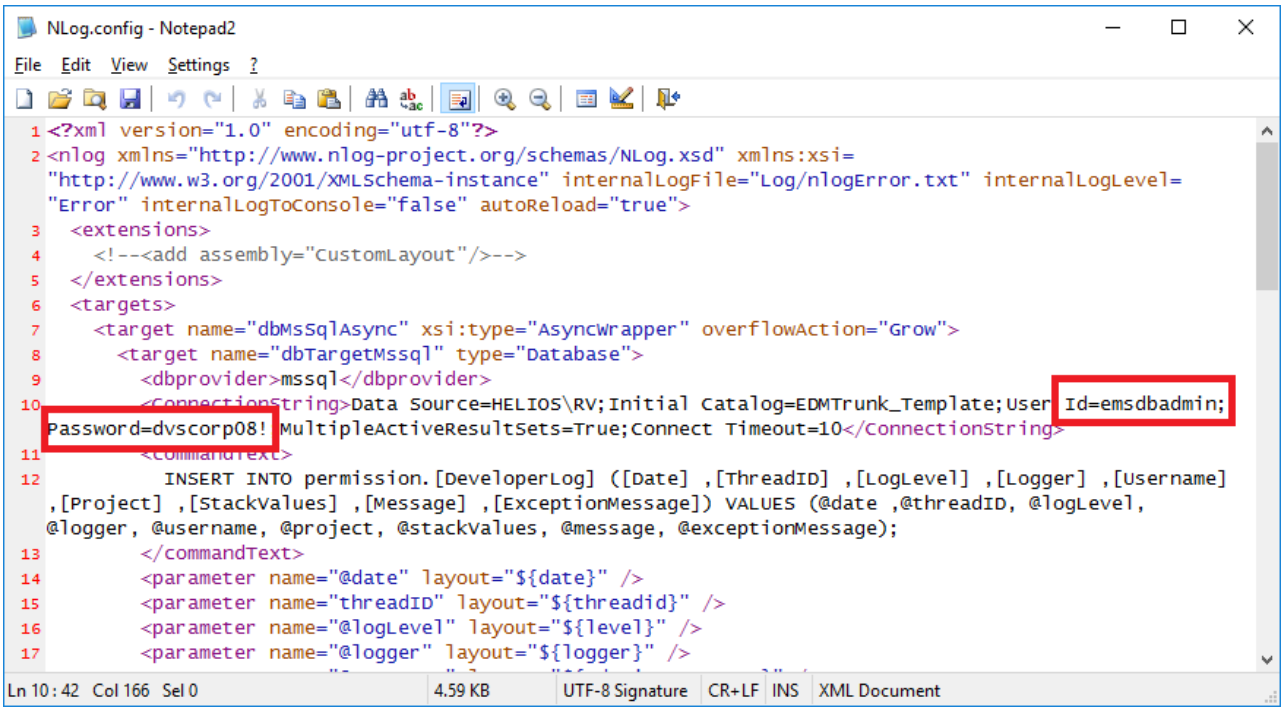
6.3.4 Credentials In Plain Text

The application has several places that included credentials in plain text hard coded within various locations and configuration files. Best practices dictate that credentials should always be encrypted whenever they are stored on the filesystem of a machine. Failing to do so can allow sensitive credentials to potentially be compromised and utilized to manipulate results. This is considered a basic security requirement that even low-risk applications should follow. The Election Management System would be considered a high-risk system.

6.3.4.1 C:\Program Files\Dominion Voting Systems\Smart Card Service\NLog.config

Database credentials are stored in plain text without any encryption within a configuration file for the Smart Card Service. The naming of this password suggests that this default password has gone unchanged since 2008. This is supported by a 2010 defect report that cited this same password as being hard coded within the application:

https://www.eac.gov/sites/default/files/voting_system/files/Deficiency%20Report.pdf



6.3.4.2 C:\Program Files\Dominion Voting Systems\Results Tally and Reporting\ DVS.DemocracySuite.ResultTally.exe.Config

The configuration for the Results Tally and Reporting Application has a location where a password would be stored in plain text. Since Antrim writes the results to the local file system rather than a network machine; this entry does not directly represent a risk to Antrim County. However, this shows a pattern of not following well recognized industry best practices.

```
  <ConnectionConfiguration AppSrvIP="emsserver" AppSrvName="emsapplicationserver"
    AppSrvPort="" DbProvider="SqlServer" LastProject="Antrim November 2020"
    LeftBrowserPath="" MirrorMode="false" MirrorServerName="dvssqlserver"
    RightBrowserPath="" SqlServerName="emsserver" USBPort="1" UseSSL="False"
    VitnesServerName="dvssqlserver" XmlFileBrowserPath="" ConnectionStringMirror="Data
Source={3};Failover Partner={4};Connect Timeout=60;Load Balance Timeout=20;initial catalog={0};user
id={1};password={2}"
    ConnectionStringStandAllone="server={3};user
id={1};password={2};MultipleActiveResultSets=True;database={0};connection
reset=false;pooling=true;enlist=true;min pool size=1;max pool size=50" />
  <TransferPointSettings>
    <TransferPoints>
      <Clear />
      <TransferPointElement DirectoryPath="C:\Users\EMSADMIN\Desktop\Election Results November 2020"
        HostName="" IsLocal="True" IsPublic="False" Name="Results Export"
        Password="" Port="" TransferPointType="Folder" Username="" />
    </TransferPoints>
  </TransferPointSettings>
```

6.3.5 C:\Program Files\Dominion Voting Systems\Results Tally and Reporting\NSLog.config

The NSLog.Config for the Results Tally and Reporting has multiple database connections hard coded within the configuration file. Giving the naming and database types listed with the connections string, it's unclear if these are currently in-use in the application. However, it further demonstrates that storing passwords in plain text is common within the application suite.

```
  <targets>
    <target name="dbMsSqlAsync" xsi:type="AsyncWrapper" overflowAction="Discard">
      <target name="dbTargetMssql" type="Database">
        <dbprovider>mssql</dbprovider>
        <ConnectionString>Data Source=HERMES\DEVTEST;Initial Catalog=Logs;User ID=logwriter;Password=logwriter;
</ConnectionString>
        <commandText>
          INSERT INTO [Logs].[dbo].[DeveloperLog] ([Date] ,[ThreadID] ,[LogLevel] ,[Logger] ,[Username] ,[Project]
,[StackValues] ,[Message] ,[ExceptionMessage]) VALUES (@date ,@threadID, @logLevel, @logger, @username, @project,
@stackValues, @message, @exceptionMessage);
        </commandText>
        <parameter name="@date" layout="${date}"/>
        <parameter name="threadID" layout="${threadid}"/>
        <parameter name="@logLevel" layout="${level}"/>
        <parameter name="@logger" layout="${logger}"/>
        <parameter name="@username" layout="${mdc:item=EMSUser}"/>
        <parameter name="@project" layout="${mdc:item=EMSProject}"/>
        <parameter name="@stackValues" layout="${stacktrace:topFrames=12}"/>
        <parameter name="@message" layout="${message}"/>
        <parameter name="@exceptionMessage" layout="${exception:format=Message, Type, ShortType, ToString, Method,
StackTrace}"/>
        <!--<parameter name="@secCode" layout="${secCode:SecCodeCalcOn=true}"/>-->
      </target>
    </target>
    <target name="dbPostgreAsync" xsi:type="AsyncWrapper" overflowAction="Discard">
      <target name="dbTargetPostgre" type="Database">
        <dbprovider>Npgsql.NpgsqlConnection, Npgsql</dbprovider>
        <ConnectionString>Server=localhost;Port=5432;User Id=postgres;Password=postgresapwd13;Database=TestDatabase;
</ConnectionString>
```

6.3.6 C:\Program Files\Dominion Voting Systems\Election Event Designer\NSLog.config

The NSLog.Config for the Election Event Designer has multiple database connections hard coded within the configuration file. Giving the naming and database types listed with the connections string, it's unclear if these are currently in-use in the application. However, it further demonstrates that storing passwords in plain text is common within the application suite.

```
  <targets>
    <target name="dbMsSqlAsync" xsi:type="AsyncWrapper" overflowAction="Discard">
      <target name="dbTargetMssql" type="Database">
        <dbprovider>mssql</dbprovider>
        <ConnectionString>Data Source=HERMES\DEVTEST;Initial Catalog=Logs;User ID=logwriter;Password=logwriter;
</ConnectionString>
        <commandText>
          INSERT INTO [Logs].[dbo].[DeveloperLog] ([Date] ,[ThreadID] ,[LogLevel] ,[Logger] ,[Username] ,[Project]
,[StackValues] ,[Message] ,[ExceptionMessage]) VALUES (@date ,@threadID, @logLevel, @logger, @username, @project,
@stackValues, @message, @exceptionMessage);
        </commandText>
        <parameter name="@date" layout="${date}"/>
        <parameter name="threadID" layout="${threadid}"/>
        <parameter name="@logLevel" layout="${level}"/>
        <parameter name="@logger" layout="${logger}"/>
        <parameter name="@username" layout="${mdc:item=EMSUser}"/>
        <parameter name="@project" layout="${mdc:item=EMSProject}"/>
        <parameter name="@stackValues" layout="${stacktrace:topFrames=12}"/>
        <parameter name="@message" layout="${message}"/>
        <parameter name="@exceptionMessage" layout="${exception:format=Message, Type, ShortType, ToString, Method,
StackTrace}"/>
        <!--<parameter name="@secCode" layout="${secCode:SecCodeCalcOn=true}"/>-->
      </target>
    </target>
    <target name="dbPostgreAsync" xsi:type="AsyncWrapper" overflowAction="Discard">
      <target name="dbTargetPostgre" type="Database">
        <dbprovider>Npgsql.NpgsqlConnection, Npgsql</dbprovider>
        <ConnectionString>Server=localhost;Port=5432;User Id=postgres;Password=postgresapwd13;Database=TestDatabase;
</ConnectionString>
```

6.3.7 C:\Program Files\Dominion Voting Systems\Election Data Translator\ DVS.Bridging.ImportAdapter.exe.config

The DVS.Bridging.ImportAdapter.exe.config for the Election Data Translator has multiple locations for credentials hard coded within the configuration file. It does not appear that Antrim utilizes this feature, so these appear to be blank. However, it further demonstrates that storing passwords in plain text is common within the application suite.

```
<appSettings>
  <add key="dvs" value="rDq6LWxe+bWypbsNj1TL0g=="/>
  <add key="dvsa" value="kh996Vk9ch6ZCjK5sp0B6Q=="/>
  <add key="remSvr" value="http://localhost/emsapplicationserverdev/RemoteDbProviderImpl.rem"/>
  <add key="remoteAdo" value="false"/>
  <add key="isIpConst" value="false"/>
  <add key="srvName" value=""/>
  <add key="adminUserName" value=""/>
  <add key="adminPassword" value=""/>
  <add key="userPassword" value=""/>
  <add key="userName" value=""/>
  <add key="dirPath" value="" />
  <add key="DbProvider" value="SqlServer"/>
  <add key="STA" value="true"/>
  <add key="BulkInsertCheckConstraints" value="true"/>
  <add key="ClientSettingsProvider.ServiceUri" value=""/>
  <add key="wcfBinding" value="net.tcp"/>
</appSettings>
```

6.4 Credential Management

Credential reuse appears to be relatively common across the organization, and across multiple deployments of the application. The password dvscorp08!, which was in use in Antrim County has showed up in prior deficiency reports, and in breach data associated with employees of the EMS vendor. Based on its naming, this password is over 12 years old and still in use today. The continued use of this password makes it easy for a potential attacker to guess a password and get into the system to manipulate data.

6.4.1 Password reuse

Reviewing the passwords utilized for Antrim County going back to August 2018; it appears that the password “dvscorp08!” has been utilized for at least one account since 2018 and in many cases that same password was utilized for most if not all the accounts.

An 2012 EAC report, “WYLE TEST REPORT NO. T57381-01APPENDIX A.11DEFICIENCY REPORT“, reported on page 10 the **dvscorp08!** Password was hardcoded into the system and first reported on 2010-08-16 14:28.

https://www.eac.gov/sites/default/files/voting_system/files/Deficiency%20Report.pdf

6.4.2 Breach Data

A search of breach data associated with the EMS vendor's domains shows regular use of the password "dvscorp08!".

2017-07-19 Breach
EMAIL | SHA-1 | CLEAR PASS
masha.REDACTED@dominionvoting.com | a02151de1fa63caca41e4904e35a3972fc824b06 | dvscorp08!

2017-12-11 Breach

masha.REDACTED@dominionvoting.com:dvscorp08!
masha.REDACTED@dvscorp.com:dvscorp08!
masha.REDACTED@gmail.com:dvscorp08!

6.5 Certification

The Election Assistance Commission's (EAC) list of approved software for the EMS does not appear to include Microsoft SQL Server Management Studio, but this software is installed on the machine. Microsoft SQL Server Management Studio is a database administration tool which makes it easy to directly edit entries within the database. This could potentially be utilized to change vote values.

Since this tool is a separate install from Microsoft SQL Server, it is our understanding that it would be required to explicitly mentioned on the list of certified software in order to be allowed to exist on an EAC certified configuration (See pages 4-13):
https://www.eac.gov/sites/default/files/voting_system/files/Dominion_Voting_Systems_D-Suite_5.5-B_Test_Plan-Rev._02.pdf

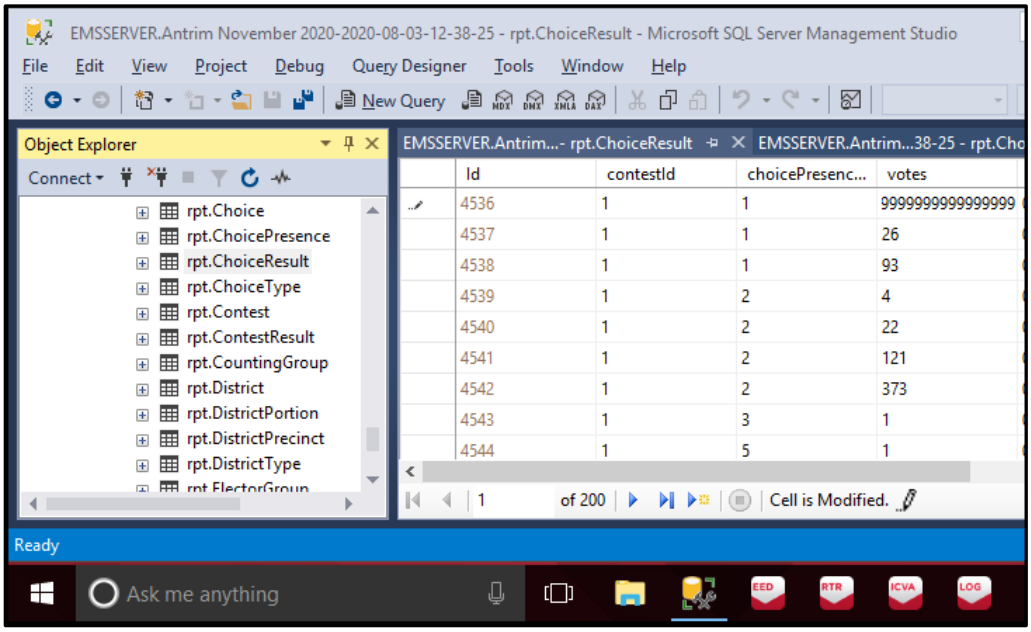


Figure 3 - SQL Management Tools allows the easy editing of a vote.

7 AFFIRMATION

I declare that I am over the age of 18, and I understand and believe in the obligations of an oath. Under penalty of perjury laws of the State of Michigan and the United States I attest that the foregoing report is true and correct, and that this report was executed this 9th day of April, 2021.

Douglas Logan

8 ABOUT CYBER NINJAS

Cyber Ninjas is an application security consulting company specializing in code review, ethical hacking, training and security program development. Our staff represents decades of experience in a variety of areas including application support, development, product management, and application security. This experience across all areas of the software development life cycle gives us a unique perspective on how to build security into your existing processes. We can help you build a software security program, expand the capabilities of your existing staff, or simply perform a security assessment of your software or your company. With everything we do, our goal is to build the knowledge within your organization. We strongly believe that "Security comes with knowledge." ™; and that it is our job as Cyber Ninjas to train and teach through every engagement in order to build up capabilities within your organization.

APPENDIX A: BIO – DOUGLAS LOGAN

Douglas Logan has handled Cyber Security for major companies and organizations around the country, such as the Federal Communications Commission, JP Morgan Chase, Bank of America, Citibank, Sally Mae, and more. In 2015, he was named a winner of the prestigious SANS 2015 Difference Makers Award.

Mr. Logan (CISSP, GWAPT, GCIH) is the CEO and Principal Consultant for Cyber Ninjas, a Sarasota, Florida-based company. Mr. Logan is responsible for working with organizations to evaluate their current cyber security risks, educate stakeholders on the nature and causes of those risks and establish policies, programs, and procedures that provide long-term protection.

Mr. Logan founded Cyber Ninjas under the mission of building organizations' cyber security capabilities by developing their people and processes, providing them with the opportunity to eventually handle their own security requirements. His solution-focused services include enterprise threat analysis and modeling, security program development, secure software development life-cycle (sSDLC) creation, malicious code detection, training, staff mentoring, code review and ethical hacking. "We believe there is no point in breaking something if you can't offer a reasonable way to fix it," he says**.**

Prior to founding Cyber Ninjas in 2013, Mr. Logan was a Senior Consultant for Cigital where, among other responsibilites, he helped launch the Bloomington, Indiana office. Under Mr. Logan's technical leadership, Cigital was able to scale their Vulnerability Assessement Managed Service in less than a year from three people conducting roughly 10 assessments a month, to about 20 individuals performing 250 assessments a month. Mr. Logan's process oriented methodology allowed him to place new hires straight out of college to billable work in under 10 days, and had those same individuals leading teams within 60 days. After a year of building people and processes, the entire system Mr. Logan built was self-propetuating and completely self-sufficient, allowing him to step into other projects.

Mr. Logan was also involved in many other areas of Cigital's business, including mobile threat modeling and threat analysis, red team enterprise risk assessments, advanced penetration testing, and instructor lead training. He is the author of Cigital's Android Penetration Testing class, and co-author and team-lead responsible for creating the iOS Penetration Testing class.

Before Cigital, Mr. Logan had 12 years combined experience in the IT field, including roles as Server Administration, Development, and Product Management.

His broad experience not only gives him a deep technical backing, but allows him to design solutions that integrate with normal day-to-day IT processes.

Outside of work Mr. Logan volunteers for the US Cyber Challenge; a non-profit organization dedicated to finding America's brightest and getting them plugged into the Cyber Security field. In that role he helps shape America's future cyber warriors to help defend our nation.

Mr. Logan holds Bachelors degrees in both Business Management and Accounting from Guilford College in Greensboro, NC.