

Iterative Security Test Report

Version: 2

Project Name: Cybersecurity Capstone One Project

Test Completed on: October 17, 2021

Team Members: Riley Dorough, Kayla Echols, Julia Wilkins, Brett Wolff

BLUF Statement: Project is on schedule, there are currently no issues blocking progress.

Vulnerability/Risk

Level:

- Low
- High
- Untested
- Medium
- Critical

Executive Summary

As we continue working on the requirements for the assigned project, the team will discover multiple risks and security concerns. There are many Common Vulnerability Exposures (CVEs) that apply to the various software's including SQL Server, openHistorian, and Grafana and the operating systems that our team is using. Our team will document the various vulnerabilities discovered and our response in how we are working towards fixing the problem. We are now moving into the testing phase of the project and out of the building phase. This will lead to more findings in the coming weeks.

CVSS v3.0 Ratings

Severity	Base Score Range
None	0.0
Low	0.1-3.9
Medium	4.0-6.9
High	7.0-8.9
Critical	9.0-10.0

Figure 1. Vulnerability Metrics. Retrieved from <https://nvd.nist.gov/vuln-metrics/cvss>.

CONFIDENTIAL

OWASP Top 10 Findings

There are a handful of risks which are applicable to the current status of the project. These are the Man in the Middle (MITM), Denial of Service/Distributed Denial of Service (DoS/DDoS), social engineering, Remote Desktop Protocol (RDP), unencrypted communication, and personal identifiable information (PII) leakage. Most of these risks threaten the network layer and thusly, are not evaluated using the manual source code review. These risks, other than social engineering, will be evaluated in the iterative testing cycle, once penetration testing and attempts to break the software begin.


For descriptions of security test cases and their associated risks and security requirements, see Appendices B and C.

Risk	Test Cases (See Appendix C – Security test Cases)	Status
Man in the Middle Attack (MITM)	TC-PT-001: Pen Test: Man in the Middle <ul style="list-style-type: none">SR-AUTN-001SR-AUTN-002SR-AUTN-003	Not Tested
DoS/DDoS (Denial of Service and Distributed Denial of Service)	TC-PT-003: Pen Test: DOS/DDOS <ul style="list-style-type: none">No related security requirements, but DOS/DDOS attacks are a risk that need to be assessed.Both Host and Public Website	Not Tested
Social Engineering	Attempts to gain sensitive information <ul style="list-style-type: none">PhishingPII Leakage	Not Tested
Windows Remote Desktop Protocol (RDP)	Attempts to gain access to the desktop remotely to gain control and access information	Not Tested
Unencrypted Communications	Data in transit could be vulnerable <ul style="list-style-type: none">If unencrypted, someone could intercept informationCould also be under MITM	Not Tested
Unfamiliar/ Insecure Protocols	OPTO devices utilize memory map protocol (MMP). Testing needs to be done to ensure its security. <ul style="list-style-type: none">None/ minimal embedded securityEasy to crack encryption methodsMITM capabilities	In Progress

CVE's

After review of the different technologies being used in this product, we decided to research different CVE's that may be applicable. While there is a handful of CVEs that are applicable to our project, most of them can be resolved simply by updating the software being used. So rather than spending time talking about such software, the focus will be on CVE's that are not so easily resolved.

This also applies to products that will be designed in house as proprietary products. They should hopefully not have any inherent CVE or vulnerabilities, but they will be tested as if they have them, both alone and integrated with the system. Some CVEs will likely exist for the platforms that the software will be built on, however this is not the focus of the software that will be developed/integrated. Namely, the Data historian or the database.

CVE	Test Cases	Status
CVE-2021-040444	Ensuring all Word files are opened only in safe mode.	 In Progress

Manual Code Review Findings

The purpose of the Manual Source Code Review is to determine areas of interest for the iterative testing cycle, as well as document and communicate the initial security posture of the project’s software. Beginning code review/ research returned large amounts of CVEs for non-open-source applications and systems

For descriptions of security test cases and their associated risks and security requirements, see Appendices B and C.

Test Cases & Associated Risks and Requirements	Discovered Vulnerabilities & Impact Summary	Assigned To:	Status
<div><div>[James Dorough]</div><div>Windows 10 21h1:</div><div><div></div><div>CVEs to be listed here</div></div></div>	None at this time to be tested. Standing LNG infrastructure to be accessible first.	N/A	<div>In</div> <div>Progress</div>

CONFIDENTIAL

Automated Code Review Findings

The team is in the process of researching different tools to review the source code for the various programs that we are using for this project including openHistorian and SQL. We anticipate having further code to review as we begin working on the .NET framework and when we begin writing in SQL to construct our databases.

For descriptions of security test cases and their associated risks and security requirements, see Appendices B and C.

Test Cases & Associated Risks and Requirements	Discovered Vulnerabilities & Impact Summary	Assigned To:	Status
<div>[Kayla Echols] SQL:<ul style="list-style-type: none">CVEs to be listed here</div>	None at this time to be tested.	N/A	In Progress

Automated Dynamic Testing Findings

Automated dynamic testing involves checking the response of the system to the application being run. It observes the behavior of the software system, memory usage, CPU usage, and overall performance of the system. The main goal of automated dynamic testing is to ensure that the finalized product is in a correct working state that does not overexert the machine to unstable levels. Automated dynamic testing can be conducted using unit testing, integration testing, and system testing. We have determined that this type of testing is not within the scope of the customer's goals for the project. At this time automated dynamic testing will not be applicable. As machine learning is incorporated into the CVALNG project, automated testing becomes much more feasible.

CONFIDENTIAL

Penetration Testing Findings

The purpose of the penetration test is to simulate ways in which an adversary attacking a network running the software would interact with and attack the software. Due to the large number of test cases in the penetration test, it will be split across multiple iterative tests to allow each case to be fully and carefully tested.

For descriptions of security test cases and their associated risks and security requirements, see Appendices B and C.

Test Cases & Associated Risks and Requirements	Discovered Vulnerabilities & Impact Summary	Assigned To:	Status
[Charity Barker] TC-PT-001: Man in the Middle <ul style="list-style-type: none"> SR-AUTN-001 SR-AUTN-002 SR-AUTN-003 	The testing and development network is not configured in a manner that allows network traffic from one machine to be collected and viewed by another. This makes it difficult to test Man in the Middle attack testing. The action item related to this test case for the upcoming week is to see if the testing environment can be reconfigured to allow collection of network traffic.		Not Tested
[Dylan Van Reenen] TC-PT-002: DNS Hijacking	The .NET framework deals with DNS resolution using a class titled System.Net which contains functions such as Resolve (resolves a DNS host name or IP address) and GetHostByName (gets the DNS information for the specified DNS host name). After review of the code, multiple source code files were found to contain the System.Net class. However, none of the source code use any of the functions dealing with DNS.	Dylan Van Reenen	Not Resolved
[Kaitlin Weathers] TC-PT-003: DOS/DDOS <ul style="list-style-type: none"> SR-ATEN-001 SR-ATEN-005 	The .NET framework contains the CompressedStack class and is defined within the CompressedStack.cs file. The CompressedStack class represents the code access security information containing a Deny action (CompressedStack Class, n.d.). The CodeAccess Permission.Deny Method to prevent callers higher in the call stack from using the code (CodeAccess Permission.Deny Method, n.d.).	Kaitlin Weathers	In Progress
[Charity Barker] TC-PT-004: API Auth Bypass <ul style="list-style-type: none"> SR-IDEN-002 	<p>After inspecting the source code, the team has decided to ask the client before testing the API. Because the API may be a production service, we'd rather not attack it without express permission and further information from the client.</p> <p>Per the client meeting on 10/11/19, we will not be given access to the API to perform testing. API Authentication security recommendations will be documented and delivered to the client. This test case will be marked as resolved once the client receives the security recommendations.</p>	Charity Barker	In Progress
[Dylan Van Reenen] TC-PT-005: Malicious Binary Delivery <ul style="list-style-type: none"> SR-INTG-001 SR-INTG-002 SR-INTG-003 SR-IMMU-001 	After review of the source code, it has been determined that the best way to conduct malicious binary injection is to create a malicious package to be installed and run by the application. Another possibility is to modify an existing package before installation. However, this will be difficult without prior knowledge on how the packages are structured. According to the file 'AgentUtilities.cs' a package is downloaded using the function 'DownloadPackageVersion' from http://....and saved as a .zip file in	Dylan Van Reenen	Not Resolved

CONFIDENTIAL

<ul style="list-style-type: none"> SR-SYSM-002 			
<p>[Dylan Van Reenen]</p> <p>TC-PT-006: Privilege Escalation</p> <ul style="list-style-type: none"> SR-IMMU-002 SR-IMMU-003 SR-SURV-001 SR-SYSM-002 SR-AUTR-001 SR-AUTR-002 	<p>The System agent code takes advantage of the System.ServiceBase class. This class utilizes "ServiceProcessInstaller" which is responsible for installing the service onto the system. "ServiceProcessInstaller" also specifies which account is going to be used to install the service on the system. In "ProjectInstaller.cs" there is a function "InitializeComponent" which specifies that the LocalSystem account will be used to install and start the process contained in the package fed into the agent. The LocalSystem account has full access to the resources on the machine (source). If a malicious package is able to be fed into the agent, we could use the LocalSystem account to our advantage.</p>		<p>Not Resolved</p>
<p>[Aaron Demers]</p> <p>TC-PT-007: COM Hijacking</p> <ul style="list-style-type: none"> SR-AUTN-002 SR-IMMU-002 SR-IMMU-003 	<p>COM is not used by the code within the scope of the project. If we end up working with the software engineering team on their package that interacts with applications, COM Hijacking will become an area of interest.</p>		<p>Resolved</p>
<p>[Heidi Waddell]</p> <p>TC-PT-008: Back Doors</p> <ul style="list-style-type: none"> SR-IDEN-002 	<p>At this time, the risk of backdoors in the code has been assessed through a manual code review. Based on the safeguards built-in to Velocity's source code, the risk of backdoors is at a low level, and therefore is not at a high priority for testing. However, penetration tests to ascertain the risk of backdoors in the code will be performed in the future.</p>		<p>Not Tested</p>

Summary Graphs

Listed CVEs Per Host Version - 2021

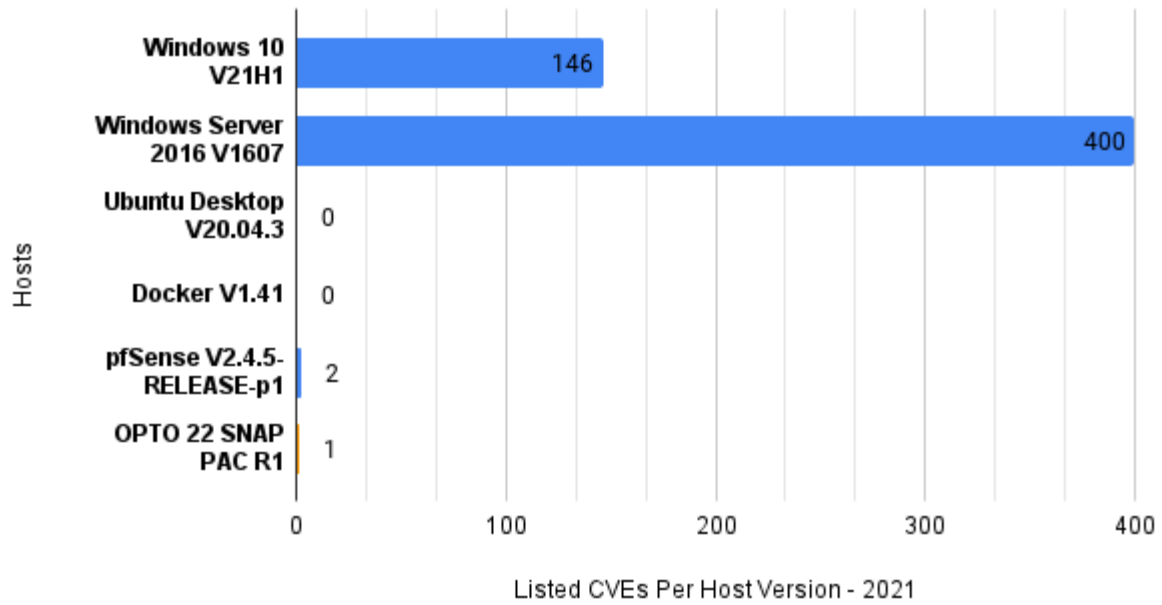


Figure 9.1 Total list of known CVEs per host version in 2021. Data Retrieved from: <https://www.cvedetails.com/>.

Action Items & Areas of Interest

As we wrap up building and move into testing it is important to take note of any potential issues we might have. After reviewing CVE's as well as surface level code reviews, we have yet to find any major issues in our system. However, this is only a surface level review. Over the next 7 weeks we will be digging more into the system to ensure that we are being thorough in our review. Below is a list of areas we are aware of and will be looking into.

- **API Authentication Bypass:**
 - Discover and investigate the local API.
 - From these discoveries, recommendations for securing these functions will be created.
- **Privilege Escalation:**
 - Perform testing and analysis to better understand the usage and delivery method of the packages that get run by the agent.
 - Further investigate the differences between the system agent and the user agent. Questions to answer include:
 - Which ones handle specific networking functions? Is there segmentation implemented?
 - If so, how and why?
 - Which one runs with elevated permissions, and what does it interface with?
- **Logging:**
 - Research and examine the implementation of a more robust system of logging and investigate the creation of a patch to apply these changes. Currently, the system of logging is decently secure, but gaps remain, and more security is feasible and within scope.
 - Currently documenting logging improvements to be discussed at the next client interaction.
- **Man in the Middle:**
 - Modify the testing environment to allow collection of network traffic.
- **Hijacking**
 - Continue to check if there is any possibility of gaining admin rights on the user system in order to redirect DNS traffic.
- **Malicious Binary Delivery**
 - Determine whether the installer code can be injected into, or if a different mode of injection is used to gain access to a modified installer.
- **SQL Injection**
 - Determine if any applications developed using .NET as well as open historian are susceptible to SQL injection.

API Development Progress

The API that we are currently working on is the development of the openHistorian which uses the OPTO22 software that contains the data from the OPTO22 hardware to communicate to a database and is read into the openHistorian. The purpose of the openHistorian is to organize the data from the OPTO22 hardware into a format that is easily readable by users. The openHistorian offers an open-source visual analytics and interactive web application which allows users to view charts, graphs, and various alerts called Grafana. We plan to gain a better understanding of Grafana as it would be helpful in allowing the information from the openHistorian to be more easily accessible and visually appealing for the users.

Security Recommendations

Security Vulnerabilities in .NET Framework 4.8

Currently, we are planning on using the .NET Framework version 4.8 to develop an application for alerting. There are seven currently identified common vulnerabilities and exposures (CVEs) in this version of the framework, none of which are critical vulnerabilities (scored above a 9). The two highest have a score of 7.5. Both of these vulnerabilities are a remote code execution vulnerability. Both the vulnerabilities primarily effect Fedora Linux distributions but also affects .NET and .NET core. As of now there is no fix to this solution, so it is recommended to be aware of the vulnerability and to await any potential updates to the framework.

The detailed summaries for the seven vulnerabilities are as follows:

#	CVE ID	CWE ID	# of Exploits	Vulnerability Type(s)	Publish Date	Update Date	Score
1	CVE-2021-34485				2021-08-12	2021-08-18	2.1
	.NET Core and Visual Studio Information Disclosure Vulnerability						
2	CVE-2021-31957			DoS	2021-06-08	2021-07-07	5.0
	ASP.NET Denial of Service Vulnerability						
3	CVE-2021-31204 269				2021-05-11	2021-07-07	4.6
	.NET and Visual Studio Elevation of Privilege Vulnerability						
4	CVE-2021-26701			Exec Code	2021-02-25	2021-08-16	7.5
	.NET Core Remote Code Execution Vulnerability This CVE ID is unique from CVE-2021-24112.						
5	CVE-2021-26423			DoS	2021-08-12	2021-08-19	5.0
	.NET Core and Visual Studio Denial of Service Vulnerability						
6	CVE-2021-24112			Exec Code	2021-02-25	2021-07-07	7.5
	.NET Core Remote Code Execution Vulnerability This CVE ID is unique from CVE-2021-26701.						
7	CVE-2021-1721			DoS	2021-02-25	2021-08-16	4.3
	.NET Core and Visual Studio Denial of Service Vulnerability						

Figure 11. Critical CVE's in the .NET Framework 4.

Acronym Definitions

OWASP – Open Web Application Security Project

CVE – Common Vulnerabilities and Exposures

CVSS – Common Vulnerability Scoring System

MITM – Man in the Middle Attack

DOS/DDOS – Denial of Service / Distributed Denial of Service

MSCR – Manual Source Code Review

ASCR – Automated Source Code Review

PT – Penetration Test, Pentest

TC – Test Case

SR – Security Requirement

LNG – Liquid Natural Gas

MMP – Memory Map Protocol

• References

NIST. (n.d.). *NIST CVE details*. NVD. Retrieved October 16, 2021, from <https://nvd.nist.gov/vuln/detail/CVE-2021-26701>.

Security vulnerabilities. CVE security vulnerability database. (n.d.). Retrieved October 13, 2021, from <https://www.cvedetails.com/vulnerability-list>.

Appendices
