****

# CS 305 Project One

**Artemis Financial Vulnerability Assessment Report**

Table of Contents

[Document Revision History 3](#_Toc32574607)

[Client 3](#_Toc32574608)

[Instructions 3](#_Toc32574609)

[Developer 4](#_Toc32574610)

[1. Interpreting Client Needs 4](#_Toc32574611)

[2. Areas of Security 4](#_Toc32574612)

[3. Manual Review 4](#_Toc32574613)

[4. Static Testing 4](#_Toc32574614)

[5. Mitigation Plan 4](#_Toc32574615)

## Document Revision History

| **Version** | **Date** | **Author** | **Comments** |
| --- | --- | --- | --- |
| 1.0 | 09/19/2021 | Pomai Ahlo | Completed All Sections |

## Client



## Developer

Pomai Ahlo

## 1. Interpreting Client Needs

Secure communications are essential for Artemis Financial, as it entails the safety of their data and the trust of their customers. As a financial company, there is a lot of sensitive data being stored, and the company values secure data handling and communication.

Consultant companies often work with global banks and insurance companies, and Artemis Financial is no different. Thus, there are going to be more international transactions in the future.

The Gramm-Leach-Bliley Act is a U.S. law that requires institutions to disclose how they share and protect private information. While most assume this act only applies to banks and insurance companies, any company that focuses on providing financial products or services need to follow this law. Thus, it’s safe to assume that Artemis Financial needs to be cognizant of this regulation and ensure its compliance.

Malicious actors are always looking to steal sensitive information. As a company that handles finances, Artemis Financial is a prime target.

Artemis Financial needs to present itself as being sleek and with the times in order to gain the trust and attention of its current and future customers. Using evolving web technologies is one way to do that, as it can make the user experience faster and more efficient. Additionally, the use of open source libraries such as the Spring Framework allows the company to save money and time by using premade code as tools. Additionally, should there be a problem with this code, Artemis Financial will not be solely responsible for fixing the problem, as they would be if they wrote all their code in house.

## 2. Areas of Security

Based on the VAPFD, the following areas of security need to be addressed in this application:

* Input Validation: Like most web applications, this one takes input. Thus, input validation is something that needs to be checked so that we can be sure that there is no way to abuse the inputs.
* APIs: Artemis Financial wants to use the Spring Framework, which can be used for API dependencies, and RESTful APIs. Because Artemis Financial is using APIs, checking that there are secure API interactions is necessary.
* Cryptography: In the CRUD files of this application, the business name is encrypted. It is therefore necessary to ensure the encryption is secure and used properly.
* Client/Server: Any web based software application has client-server communication. A server, in this case, the code base, execute the logic for requests made by a client (through the REST API). It’s important to ensure that requests cannot be manipulated in a way that grants unauthorized access to information.
* Code Error: not only should we check for logical errors in the code, but we should also ensure that errors are being handled properly. This means providing the user with enough feedback to correct their mistakes, but without disclosing so much information that the system is compromised.
* Code Quality: This is a category that also encompasses input validation and APIs. This should always be checked, as this is where many vulnerabilities lie. If code quality is not verified, attacks such as denial of service may be possible.

## 3. Manual Review

After a manual review of the code, the following errors/vulnerabilities have been discovered:

CRUDController.java

* When you input no parameters, there is no error handling- it brings you to the “Whitelabel Error Page”, which is default for having no error page.
* There is no input validation for the parameter *business\_name*.

GreetingController.java

* Greeting does not validate input in any way.

myDateTime.java

* setMyDateTime does not set the date and time as expected. The method is empty
* retrieveDateTime returns a blank integer array when it’s supposed to return an integer array with mySecond, myMinute, and myHour.

DocData.java

* The database parameters (test, root, root) are hardcoded, which is not secure.
* There is a comment that discloses the username and password.

## 4. Static Testing

The following screenshot is of the summary of the dependency check.



Below are the dependencies and vulnerabilities.

Dependency: bcprov-jdk15on-1.46.jar

* CVE-2013-1624
  + The TLS implementation in the Bouncy Castle Java library before 1.48 and C# library before 1.8 does not properly consider timing side-channel attacks on a noncompliant MAC check operation during the processing of malformed CBC padding, which allows remote attackers to conduct distinguishing attacks and plaintext-recovery attacks via statistical analysis of timing data for crafted packets, a related issue to CVE-2013-0169.
* CVE-2015-6644
  + An information disclosure vulnerability in Bouncy Castle could enable a local malicious application to gain access to user’s private information
* CVE-2015-7940
  + The Bouncy Castle Java library before 1.51 does not validate a point is withing the elliptic curve, which makes it easier for remote attackers to obtain private keys via a series of crafted elliptic curve Diffie Hellman (ECDH) key exchanges, aka an "invalid curve attack."
* CVE-2016-1000338
  + In Bouncy Castle JCE Provider version 1.55 and earlier the DSA does not fully validate ASN.1 encoding of signature on verification. It is possible to inject extra elements in the sequence making up the signature and still have it validate, which in some cases may allow the introduction of 'invisible' data into a signed structure.
* CVE-2016-1000339
  + In the Bouncy Castle JCE Provider version 1.55 and earlier the primary engine class used for AES was AESFastEngine. Due to the highly table driven approach used in the algorithm it turns out that if the data channel on the CPU can be monitored the lookup table accesses are sufficient to leak information on the AES key being used. There was also a leak in AESEngine although it was substantially less. AESEngine has been modified to remove any signs of leakage (testing carried out on Intel X86-64) and is now the primary AES class for the BC JCE provider from 1.56. Use of AESFastEngine is now only recommended where otherwise deemed appropriate.
* CVE-2016-1000341
  + In the Bouncy Castle JCE Provider version 1.55 and earlier DSA signature generation is vulnerable to timing attack. Where timings can be closely observed for the generation of signatures, the lack of blinding in 1.55, or earlier, may allow an attacker to gain information about the signature's k value and ultimately the private value as well.
* CVE-2016-1000342
  + In the Bouncy Castle JCE Provider version 1.55 and earlier ECDSA does not fully validate ASN.1 encoding of signature on verification. It is possible to inject extra elements in the sequence making up the signature and still have it validate, which in some cases may allow the introduction of 'invisible' data into a signed structure.
* CVE-2016-1000343
  + In the Bouncy Castle JCE Provider version 1.55 and earlier the DSA key pair generator generates a weak private key if used with default values. If the JCA key pair generator is not explicitly initialized with DSA parameters, 1.55 and earlier generates a private value assuming a 1024 bit key size. In earlier releases this can be dealt with by explicitly passing parameters to the key pair generator.
* CVE-2016-1000344
  + In the Bouncy Castle JCE Provider version 1.55 and earlier the DHIES implementation allowed the use of ECB mode. This mode is regarded as unsafe and support for it has been removed from the provider
* CVE-2016-1000345
  + In the Bouncy Castle JCE Provider version 1.55 and earlier the DHIES/ECIES CBC mode vulnerable to padding oracle attack. For BC 1.55 and older, in an environment where timings can be easily observed, it is possible with enough observations to identify when the decryption is failing due to padding.
* CVE-2016-1000346
  + In the Bouncy Castle JCE Provider version 1.55 and earlier the other party DH public key is not fully validated. This can cause issues as invalid keys can be used to reveal details about the other party's private key where static Diffie-Hellman is in use. As of release 1.56 the key parameters are checked on agreement calculation.
* CVE-2016-1000352
  + In the Bouncy Castle JCE Provider version 1.55 and earlier the ECIES implementation allowed the use of ECB mode. This mode is regarded as unsafe and support for it has been removed from the provider.
* CVE-2017-13098
  + BouncyCastle TLS prior to version 1.0.3, when configured to use the JCE (Java Cryptography Extension) for cryptographic functions, provides a weak Bleichenbacher oracle when any TLS cipher suite using RSA key exchange is negotiated. An attacker can recover the private key from a vulnerable application. This vulnerability is referred to as "ROBOT."
* CVE-2018-1000613
  + Legion of the Bouncy Castle Legion of the Bouncy Castle Java Cryptography APIs 1.58 up to but not including 1.60 contains a CWE-470: Use of Externally-Controlled Input to Select Classes or Code ('Unsafe Reflection') vulnerability in XMSS/XMSS^MT private key deserialization that can result in Deserializing an XMSS/XMSS^MT private key can result in the execution of unexpected code. This attack appear to be exploitable via A handcrafted private key can include references to unexpected classes which will be picked up from the class path for the executing application. This vulnerability appears to have been fixed in 1.60 and later.
* CVE-2018-5382
  + The default BKS keystore use an HMAC that is only 16 bits long, which can allow an attacker to compromise the integrity of a BKS keystore. Bouncy Castle release 1.47 changes the BKS format to a format which uses a 160 bit HMAC instead. This applies to any BKS keystore generated prior to BC 1.47. For situations where people need to create the files for legacy reasons a specific keystore type "BKS-V1" was introduced in 1.49. It should be noted that the use of "BKS-V1" is discouraged by the library authors and should only be used where it is otherwise safe to do so, as in where the use of a 16 bit checksum for the file integrity check is not going to cause a security issue in itself.
* CVE-2020-26939
  + In Legion of the Bouncy Castle BC before 1.61 and BC-FJA before 1.0.1.2, attackers can obtain sensitive information about a private exponent because of Observable Differences in Behavior to Error Inputs. This occurs in org.bouncycastle.crypto.encodings.OAEPEncoding. Sending invalid ciphertext that decrypts to a short payload in the OAEP Decoder could result in the throwing of an early exception, potentially leaking some information about the private exponent of the RSA private key performing the encryption.

Dependency: log4j-api-2.12.1.jar

* CVE-2020-9488
  + Improper validation of certificate with host mismatch in Apache Log4j SMTP appender. This could allow an SMTPS connection to be intercepted by a man-in-the-middle attack which could leak any log messages sent through that appender.
  + Fixed in version 2.13.2
  + Previous versions can set the system property mail.smtp.ssl.checkserveridentity to true to globally enable hostname verification for SMTPS connections

Dependency: snakeyaml-1.25.jar

* CVE-2017-18640
  + The Alias feature in SnakeYAML 1.18 allows entity expansion during a load operation, a related issue to CVE-2003-1564
  + Mitigation: update

Dependency: jackson-databind-2.10.2.jar

* CVE-2020-25649
  + A flaw was found in FasterXML Jackson Databind, where it did not have entity expansion secured properly. This flaw allows vulnerability to XML external entity (XXE) attacks. The highest threat from this vulnerability is data integrity.
  + Fixed In Version: jackson-databind-2.11.0, jackson-databind-2.10.5.1

Dependency: tomcat-embed-core-9.0.30.jar

* CVE-2019-17569
  + The refactoring present in Apache Tomcat 9.0.28 to 9.0.30, 8.5.48 to 8.5.50 and 7.0.98 to 7.0.99 introduced a regression. The result of the regression was that invalid Transfer-Encoding headers were incorrectly processed leading to a possibility of HTTP Request Smuggling if Tomcat was located behind a reverse proxy that incorrectly handled the invalid Transfer-Encoding header in a particular manner. Such a reverse proxy is considered unlikely.
* CVE-2020-11996
  + A specially crafted sequence of HTTP/2 requests sent to Apache Tomcat 10.0.0-M1 to 10.0.0-M5, 9.0.0.M1 to 9.0.35 and 8.5.0 to 8.5.55 could trigger high CPU usage for several seconds. If a sufficient number of such requests were made on concurrent HTTP/2 connections, the server could become unresponsive.
* CVE-2020-13934
  + An h2c direct connection to Apache Tomcat 10.0.0-M1 to 10.0.0-M6, 9.0.0.M5 to 9.0.36 and 8.5.1 to 8.5.56 did not release the HTTP/1.1 processor after the upgrade to HTTP/2. If a sufficient number of such requests were made, an OutOfMemoryException could occur leading to a denial of service.
* CVE-2020-13935
  + The payload length in a WebSocket frame was not correctly validated in Apache Tomcat 10.0.0-M1 to 10.0.0-M6, 9.0.0.M1 to 9.0.36, 8.5.0 to 8.5.56 and 7.0.27 to 7.0.104. Invalid payload lengths could trigger an infinite loop. Multiple requests with invalid payload lengths could lead to a denial of service.
* CVE-2020-13943
  + If an HTTP/2 client connecting to Apache Tomcat 10.0.0-M1 to 10.0.0-M7, 9.0.0.M1 to 9.0.37 or 8.5.0 to 8.5.57 exceeded the agreed maximum number of concurrent streams for a connection (in violation of the HTTP/2 protocol), it was possible that a subsequent request made on that connection could contain HTTP headers - including HTTP/2 pseudo headers - from a previous request rather than the intended headers. This could lead to users seeing responses for unexpected resources
* CVE-2020-17527
  + While investigating bug 64830 it was discovered that Apache Tomcat 10.0.0-M1 to 10.0.0-M9, 9.0.0-M1 to 9.0.39 and 8.5.0 to 8.5.59 could re-use an HTTP request header value from the previous stream received on an HTTP/2 connection for the request associated with the subsequent stream. While this would most likely lead to an error and the closure of the HTTP/2 connection, it is possible that information could leak between requests.
* CVE-2020-1935
  + In Apache Tomcat 9.0.0.M1 to 9.0.30, 8.5.0 to 8.5.50 and 7.0.0 to 7.0.99 the HTTP header parsing code used an approach to end-of-line parsing that allowed some invalid HTTP headers to be parsed as valid. This led to a possibility of HTTP Request Smuggling if Tomcat was located behind a reverse proxy that incorrectly handled the invalid Transfer-Encoding header in a particular manner. Such a reverse proxy is considered unlikely.
* CVE-2020-1938
  + When using the Apache JServ Protocol (AJP), care must be taken when trusting incoming connections to Apache Tomcat. Tomcat treats AJP connections as having higher trust than, for example, a similar HTTP connection. If such connections are available to an attacker, they can be exploited in ways that may be surprising. In Apache Tomcat 9.0.0.M1 to 9.0.0.30, 8.5.0 to 8.5.50 and 7.0.0 to 7.0.99, Tomcat shipped with an AJP Connector enabled by default that listened on all configured IP addresses. It was expected (and recommended in the security guide) that this Connector would be disabled if not required. This vulnerability report identified a mechanism that allowed: - returning arbitrary files from anywhere in the web application - processing any file in the web application as a JSP Further, if the web application allowed file upload and stored those files within the web application (or the attacker was able to control the content of the web application by some other means) then this, along with the ability to process a file as a JSP, made remote code execution possible. It is important to note that mitigation is only required if an AJP port is accessible to untrusted users. Users wishing to take a defence-in-depth approach and block the vector that permits returning arbitrary files and execution as JSP may upgrade to Apache Tomcat 9.0.31, 8.5.51 or 7.0.100 or later. A number of changes were made to the default AJP Connector configuration in 9.0.31 to harden the default configuration. It is likely that users upgrading to 9.0.31, 8.5.51 or 7.0.100 or later will need to make small changes to their configurations.
* CVE-2020-8022
  + A Incorrect Default Permissions vulnerability in the packaging of tomcat on SUSE Enterprise Storage 5, SUSE Linux Enterprise Server 12-SP2-BCL, SUSE Linux Enterprise Server 12-SP2-LTSS, SUSE Linux Enterprise Server 12-SP3-BCL, SUSE Linux Enterprise Server 12-SP3-LTSS, SUSE Linux Enterprise Server 12-SP4, SUSE Linux Enterprise Server 12-SP5, SUSE Linux Enterprise Server 15-LTSS, SUSE Linux Enterprise Server for SAP 12-SP2, SUSE Linux Enterprise Server for SAP 12-SP3, SUSE Linux Enterprise Server for SAP 15, SUSE OpenStack Cloud 7, SUSE OpenStack Cloud 8, SUSE OpenStack Cloud Crowbar 8 allows local attackers to escalate from group tomcat to root.
  + This issue affects SUSE Enterprise Storage 5 tomcat versions prior to 8.0.53-29.32.1. SUSE Linux Enterprise Server 12-SP2-BCL tomcat versions prior to 8.0.53-29.32.1. SUSE Linux Enterprise Server 12-SP2-LTSS tomcat versions prior to 8.0.53-29.32.1. SUSE Linux Enterprise Server 12-SP3-BCL tomcat versions prior to 8.0.53-29.32.1. SUSE Linux Enterprise Server 12-SP3-LTSS tomcat versions prior to 8.0.53-29.32.1. SUSE Linux Enterprise Server 12-SP4 tomcat versions prior to 9.0.35-3.39.1. SUSE Linux Enterprise Server 12-SP5 tomcat versions prior to 9.0.35-3.39.1. SUSE Linux Enterprise Server 15-LTSS tomcat versions prior to 9.0.35-3.57.3. SUSE Linux Enterprise Server for SAP 12-SP2 tomcat versions prior to 8.0.53-29.32.1. SUSE Linux Enterprise Server for SAP 12-SP3 tomcat versions prior to 8.0.53-29.32.1. SUSE Linux Enterprise Server for SAP 15 tomcat versions prior to 9.0.35-3.57.3. SUSE OpenStack Cloud 7 tomcat versions prior to 8.0.53-29.32.1. SUSE OpenStack Cloud 8 tomcat versions prior to 8.0.53-29.32.1. SUSE OpenStack Cloud Crowbar 8 tomcat versions prior to 8.0.53-29.32.1.
* CVE-2020-9484
  + When using Apache Tomcat versions 10.0.0-M1 to 10.0.0-M4, 9.0.0.M1 to 9.0.34, 8.5.0 to 8.5.54 and 7.0.0 to 7.0.103 if a) an attacker is able to control the contents and name of a file on the server; and b) the server is configured to use the PersistenceManager with a FileStore; and c) the PersistenceManager is configured with sessionAttributeValueClassNameFilter="null" (the default unless a SecurityManager is used) or a sufficiently lax filter to allow the attacker provided object to be deserialized; and d) the attacker knows the relative file path from the storage location used by FileStore to the file the attacker has control over; then, using a specifically crafted request, the attacker will be able to trigger remote code execution via deserialization of the file under their control. Note that all of conditions a) to d) must be true for the attack to succeed.
* CVE-2021-24122
  + When serving resources from a network location using the NTFS file system, Apache Tomcat versions 10.0.0-M1 to 10.0.0-M9, 9.0.0.M1 to 9.0.39, 8.5.0 to 8.5.59 and 7.0.0 to 7.0.106 were susceptible to JSP source code disclosure in some configurations. The root cause was the unexpected behaviour of the JRE API File.getCanonicalPath() which in turn was caused by the inconsistent behaviour of the Windows API (FindFirstFileW) in some circumstances.
* CVE-2021-25122
  + When responding to new h2c connection requests, Apache Tomcat versions 10.0.0-M1 to 10.0.0, 9.0.0.M1 to 9.0.41 and 8.5.0 to 8.5.61 could duplicate request headers and a limited amount of request body from one request to another meaning user A and user B could both see the results of user A's request.
* CVE-2021-25329
  + The fix for CVE-2020-9484 was incomplete. When using Apache Tomcat 10.0.0-M1 to 10.0.0, 9.0.0.M1 to 9.0.41, 8.5.0 to 8.5.61 or 7.0.0. to 7.0.107 with a configuration edge case that was highly unlikely to be used, the Tomcat instance was still vulnerable to CVE-2020-9494. Note that both the previously published prerequisites for CVE-2020-9484 and the previously published mitigations for CVE-2020-9484 also apply to this issue.
* CVE-2021-30640
  + A vulnerability in the JNDI Realm of Apache Tomcat allows an attacker to authenticate using variations of a valid user name and/or to bypass some of the protection provided by the LockOut Realm. This issue affects Apache Tomcat 10.0.0-M1 to 10.0.5; 9.0.0.M1 to 9.0.45; 8.5.0 to 8.5.65.
* CVE-2021-33037
  + Apache Tomcat 10.0.0-M1 to 10.0.6, 9.0.0.M1 to 9.0.46 and 8.5.0 to 8.5.66 did not correctly parse the HTTP transfer-encoding request header in some circumstances leading to the possibility to request smuggling when used with a reverse proxy. Specifically: - Tomcat incorrectly ignored the transfer encoding header if the client declared it would only accept an HTTP/1.0 response; - Tomcat honoured the identify encoding; and - Tomcat did not ensure that, if present, the chunked encoding was the final encoding.

Dependency: hibernate-validator-6.0.18.Final.jar

* CVE-2020-10693
  + A flaw was found in Hibernate Validator version 6.1.2.Final. A bug in the message interpolation processor enabled EL expressions to be evaluated as if they were valid. This flaw allows attackers to bypass input sanitization controls that developers may have put in place when using user-controlled data in error messages.
  + Fixed in: hibernate-validator 7.0.0.Alpha2, hibernate-validator 6.1.5.Final, hibernate-validator 6.0.20.Final

Dependency: spring-core-5.2.3.RELEASE.jar

* CVE-2020-5421
  + In Spring Framework versions 5.2.0 - 5.2.8, 5.1.0 - 5.1.17, 5.0.0 - 5.0.18, 4.3.0 - 4.3.28, and older unsupported versions, the protections against RFD attacks from CVE-2015-5211 may be bypassed depending on the browser used through the use of a jsessionid path parameter.
* CVE-2021-22118
  + In Spring Framework, versions 5.2.x prior to 5.2.15 and versions 5.3.x prior to 5.3.7, a WebFlux application is vulnerable to a privilege escalation: by (re)creating the temporary storage directory, a locally authenticated malicious user can read or modify files that have been uploaded to the WebFlux application, or overwrite arbitrary files with multipart request data.

Dependency: spring-jcl-5.2.3.RELEASE.jar

* CVE-2020-5421
  + This is a duplicate, it’s in the above dependency

## 5. Mitigation Plan

After interpreting your results from the manual review and static testing, identify the steps to remedy the identified security vulnerabilities for Artemis Financial’s software application. (20XX).

* When the spring framework boot is set to 2.5.4, and bouncy-castle is set to the newest version, 1.69, the dependency check shows no vulnerable dependencies. Thus, spring framework boot should be set to 2.5.4 or newer and bouncy-castle should be set to 1.69 (or newer, if applicable).
* Create an error page for if users input no business\_name that gives feedback. For example "Error. Please enter a value for business\_name".
* Validate input for the business\_name. This includes limiting the length of the input.
* Also validate input for name in the Greeting Controller. This includes blacklisting most special characters and limiting the length of the input.
* Finish setMyDateTime method: it should set mySecond, myMinute, and myHour to seconds, minutes, and hour respectively.
* Fix retrieveDateTime method: it should return an array with elements mySecond, myMinute, and myHour.
* Use a Java properties file to store database parameters instead of hardcoding them into the codebase.
* Remove comment disclosing the username and password.

**References**

Connecting to MySQL Using JDBC Driver. (n.d.). https://www.mysqltutorial.org/connecting-to-mysql-

using-jdbc-driver/

Coos, A. (March 2019) *All You Need To Know about GLBA Compliance.*

https://www.endpointprotector.com/blog/glba-the-gramm-leach-bliley-

act/#:~:text=The%20general%20tendency%20is%20to,falls%20under%20the%20Act's%20incidence.

*CVE-2013-1624 Detail*. (2013). NIST.

https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2013-1624

*CVE-2015-6644 Detail*. (2015). NIST.

https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2015-6644

*CVE-2015-7940 Detail*. (2015). NIST.

https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2015-7940

*CVE-2016-1000338 Detail*. (2016). NIST.

https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000338

*CVE-2016-1000339 Detail*. (2016). NIST.

https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000339

*CVE-2016-1000341 Detail*. (2016). NIST.

https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000341

*CVE-2016-1000342 Detail*. (2016). NIST.

https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000342

*CVE-2016-1000343 Detail*. (2016). NIST.

https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000343

*CVE-2016-1000344 Detail*. (2016). NIST.

https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000344

*CVE-2016-1000345 Detail*. (2016). NIST.

https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000345

*CVE-2016-1000346 Detail*. (2016). NIST.

https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000346

*CVE-2016-1000352 Detail*. (2016). NIST.

https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2016-1000352

*CVE-2017-13098 Detail*. (2017). NIST.

https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2017-13098

*CVE-2017-18640 Detail*. (2017). NIST.

https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2017-18640

*CVE-2018-1000613 Detail*. (2018). NIST.

https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2018-1000613

*CVE-2018-5382 Detail*. (2018). NIST.

https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2018-5382

*CVE-2019-17569 Detail*. (2019). NIST.

https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2019-17569

*CVE-2020-10693 Detail*. (2020). NIST.

https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-10693

*CVE-2020-11996 Detail*. (2020). NIST.

https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-11996

*CVE-2020-13934 Detail*. (2020). NIST.

https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-13934

*CVE-2020-13935 Detail*. (2020). NIST.

https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-13935

*CVE-2020-13943 Detail*. (2020). NIST.

https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-13943

*CVE-2020-17527 Detail*. (2020). NIST.

https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-17527

*CVE-2020-1935 Detail*. (2020). NIST.

https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-1935

*CVE-2020-1938 Detail*. (2020). NIST.

https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-1938

*CVE-2020-25649 Detail*. (2020). NIST.

https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-25649

*CVE-2020-26939 Detail*. (2020). NIST.

https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-26939

*CVE-2020-5421 Detail*. (2020). NIST.

https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-5421

*CVE-2020-8022 Detail*. (2020). NIST.

https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-8022

*CVE-2020-9484 Detail*. (2020). NIST.

https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-9484

*CVE-2020-9488 Detail*. (2020). NIST.

https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2020-9488

*CVE-2021-22118 Detail*. (2021). NIST.

https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-22118

*CVE-2021-24122 Detail*. (2021). NIST.

https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-24122

*CVE-2021-25122 Detail*. (2021). NIST.

https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-25122

*CVE-2021-25329 Detail*. (2021). NIST.

https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-25329

*CVE-2021-30640 Detail*. (2021). NIST.

https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-30640

*CVE-2021-33037 Detail*. (2021). NIST.

https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2021-33037