Sample Company Ltd.

Report for Sample Application Binary Application Pentest

January 25, 2024

Document Details

Table of Contents:

The following gives detailed information about the structure of the document:

Document History:

The following gives an overview of the release history of this document:

Table 1 - Document History

Release Date	Description	Author
January 24, 2024	Initial Release	Naunet John
January 25, 2024	Revision 1	Naunet Jane

Distribution List:

The below table contains the list of client employees who are the designated recipients of this document:

Table 2 – Distribution List

Name	E-mail Address
John Doe	john.doe@client.com
Jane Doe	jane.doe@client.com

Restrictions and Legal

This document and the information contained within is confidential and provided solely for Sample Company Ltd's informational purposes and internal use.

This document is not intended to be and should not be used by any person or entity other than Sample Company Ltd. without the explicit written permission of Naunet.

Sample Company Ltd. may choose to share the information contained in this report with auditors of relevant compliance frameworks under non-disclosure agreement.

Any unauthorized copying, distribution, and publishing, or dissemination of the information including sharing with any third party is strictly prohibited.

These materials and the information contained herein are provided by Naunet and are intended to provide general information on a particular subject or subjects and are not an exhaustive treatment of such subjects.

Accordingly, the information in these materials is not intended to constitute any professional advice or services. The information is not intended to be relied upon as the sole basis for any decision which may affect Sample Company Ltd's business.

These materials and the information contained therein are provided as is, and Naunet makes no express or implied representations or warranties regarding these materials, or the information contained therein. Without limiting the foregoing, Naunet does not warrant that the materials or information contained therein will be error-free or will meet any criteria of performance or quality. Naunet expressly disclaims all implied warranties, including, without limitation, warranties of merchantability, title, fitness for a particular purpose, non-infringement, compatibility, security, and accuracy.

Sample Company Ltd's use of these materials and information contained therein is at your own risk, and you assume full responsibility and risk of loss resulting from the use thereof. Naunet will not be liable for any special, indirect, incidental, consequential, or punitive damages or any other damages whatsoever, whether in an action of contract, statute, tort (including, without limitation, negligence), or otherwise, relating to the use of these materials or the information contained therein.

Differently from the above written in case the information and materials are expressly provided as final performance of a contract concluded between Sample Company Ltd and Naunet, Naunet takes liability that the service has been provided and the product - if any - has been prepared contractually. Naunet excludes all liability for damages arising out of or in connection with the documents, materials, information, and data provided by Sample Company Ltd. For all the questions not ruled herein, the related contract shall be applicable.

In this document Sample Company Ltd. may also be referred to as "Sample Company" or the "company" or in context as the "client", as well as VoidSec Kft. may also be referred to as "Naunet".

If any of the foregoing is not fully enforceable for any reason, the remainder shall nonetheless continue to apply completely.

VoidSec Kft. Contact Information:
info@Naunet.org
Hungary 2903
Komárom
Tamási Áron utca 4.

Assignment Summary

Assignment Summary

Sample Company Ltd. engaged VoidSec Kft. for its services to perform a binary application information security penetration test of the “Sample Application”.

The primary objective of the assessment was to identify vulnerabilities or configuration weaknesses, which could lead to unauthorized access or data theft, with a focus on finding weaknesses that are available to anyone with network-level access to the application.

Following the identification of vulnerabilities during this engagement, Sample Company is advised to implement appropriate measures to remediate these vulnerabilities and enhance the overall security posture of their application and the associated network infrastructure.

Assignment Disclaimer

We performed our testing from January 2, 2024, to January 6, 2024, and afterwards, we created this report and finalized the supporting documentation. Consequently, the findings presented in this report reflect the security status of the tested target systems and applications as of the last day of the testing period and should be treated as a snapshot in time.

Management Summary

The scope of the assessment was defined as the following:

- **Binary Application Penetration Test** for the UAT (User Acceptance Test) version of **Sample Application** binary application.

During the Binary Application Penetration Test two high-risk issues were identified.

Insecure Application Architecture: During our test we identified that the application was designed in a two-tier architecture concept, where the clients access the backend "DB01" database directly. An attacker can bypass authorization controls and privilege checks. This can allow an attacker to connect directly to the database and access or modify sensitive data.

Recommendation: This kind of vulnerability lies in the architecture concept of two-layered applications and thus cannot be completely mitigated without changing the application itself. For more detailed information, please see the finding’s recommendation section.

Unsafe Object Deserialization: The client communicated with the server by sending and receiving serialized objects. These objects were deserialized without any checks before the execution of their designated method. By exploiting this, an attacker might be able to perform remote code execution on the server.

Recommendation: We recommend using data structures (e.g., JSON) for data exchange and function calls instead of serialization.

Assignment Conclusion

We recommend mitigating the misconfigurations described in this report and performing a re-test to validate whether applied fixes are effective.

The detailed description of all found issues, the associated risks, and the recommendations to reduce the risks can be found in the following parts of this document.

Assignment Details

Assignment Background

Naunet was tasked to conduct a security assessment, which included a binary application penetration test of UAT version of the Sample Application binary application. Confidentiality, integrity, and availability of IT assets play a vital role in the business model and processes of Sample Company.

Assignment Goal

The project goal was to identify any existing security vulnerabilities, which might be exploitable by an attacker with network-level access such as a malicious employee.

Scope of the Binary Application Penetration Test

The following gives an overview specified as the scope of the Binary Application Penetration Test.

The following application have been designated as the scope:

Table 3 - Scope of the Binary Application Penetration Test

Name and version	Environment
Sample Application 1.0	UAT

The tested application used the UAT environment available on the intranet with authentication required. The functionality of the application is to allow users to manage scenarios for financial reporting.

Two test user accounts were registered for the purposes of the test, the following table contains the account details:

Table 4 - Used test accounts

Username	User Role
Username1	Administrator
Username2	Low-Privilege Role

Performed Services

The following gives an overview of the performed services:

- **Binary Application Penetration Test**, where the assignment was addressing the “Sample Application” binary application. The scan included a vulnerability assessment of the binary application with manual and semi-manual methods in a grey box approach.

Methodology

While conducting the assignment we applied our testing methodology, aligned with the NIST recommendations outlined for penetration testing (NIST Special Publication 800-115, NIST Special Publication 800-12 Revision 1), as well as processes outlined by various publications made by ENISA.

We have performed all typical tests (where applicable), including but not limited to the following:

- Configuration and Deployment Management Testing
- Identity Management Testing
- Authentication Testing
- Authorization Testing
- Session Management Testing
- Input and Data Validation Testing
- Testing for Error Handling
- Testing for Weak Cryptography
- Business Logic Testing
- Client-Side Testing
- Memory Handling Testing
- Cryptography and Data Storage Testing
- Permissions and Access Control Testing
- Network Communication Testing
- Third-Party Library Testing
- Code Obfuscation and Anti-Debugging Testing

During the assignment no attempt was made to perform the necessary tests in a covert manner.

The performed testing does not necessitate any cleanup on the target systems in scope of the assignment.

For more information on the limitations and reasoning behind not performed test types please see the following section.

The list of applied tools for the analysis can be found in the Testing Infrastructure and Tool Details section of this document.

Limitations Faced During the Test

The following list gives an overview of the limitations faced during the project:

- The scope of the penetration test did not include the following functions of the application: “Function1”, “Function2”, “Function3”, ”Function4” and “Function5”, the aforementioned functionalities were out of scope.

Findings of the Binary Application Penetration Test:

The following table gives an overview of our findings of the Binary Application Penetration Test.

Table 5 - Findings of the Binary Application Penetration Test

Finding ID	Name	Issue Type	Vulnerability Type	Risk	Status
BIN-1	Insecure Application Architecture	Application	Cross-Site Scripting (XSS)	Critical	Open
BIN-2	Unsafe Object Deserialization	Application	Security Misconfiguration	High	Open
BIN-3	Lack of Code Obfuscation	Application	Sensitive Information Disclosure	Low	Open

Detailed Findings

The following section contains the detailed technical descriptions of the misconfiguration or security weakness related findings of the assignment.

BIN-1: Insecure Application Architecture

Targets	Finding Status	Overall Risk	Critical
Sample Application v1.0	Open	Impact	High
		Likelihood	High

Description

We found that the 'Sample' application' was designed in a two-tier (server <-> client) architecture concept, where the clients access the backend "DB01" database directly, without using a third application layer in-between, as the following code excerpt shows:

```
string connectionString = @"uid=sampleUser;pwd=S[REDACTED]d1;
                            database=DB01;
                            server=targetapplication.local";
SqlConnection con = new SqlConnection(connectionString);
```

The client applications used "Windows authentication" to connect to the database and an SQL application role (protected by a password) to acquire execute privileges on stored procedures and read and write privilege on numerous tables in the DB01" database.

Impact

Due to the nature of Two-tier architecture and the observed environment configuration, a domain user could connect directly to the database, access or modify sensitive data (e.g. patient information, treatment schedules, passwords, authorization keys stored in the database, etc.).

As a result, an attacker can bypass authorization controls and privilege checks, which is only enforced in Sample Application client application. This greatly increased the attack surface of the software solution, where all data and functionality exposed by the database (due to the direct access) is a potential attack vector because the attacker can take full advantage of the privileges provided.

The attacker could also attempt to initiate further attacks trough the database software (e.g. common misconfigurations, old software versions with known vulnerabilities, etc.).

Likelihood

The likelihood of this attack is dependent on the environment in which the attacker has access to the applications and its infrastructure components.

Recommendation

This kind of vulnerability lies in the architecture concept of two-layered applications and thus cannot be completely mitigated without changing the application itself.

Due to the critical nature of the data and importance of data integrity, we recommend reworking the whole application to use a three-tier architecture, where the business logic along with the authorization and authentication logic is implemented on an application server. The client side should not be trusted to this extent as the current design allows, and not have direct access to the database. Introducing a third tier would allow to create separate security zones to assure that components are protected according to their criticality level. We note that these recommendations require significant changes is the architecture and might require changes in the database design as well.

If the complete rework of the application is not possible, Sample Company should consider developing a restricted terminal server architecture, to implement an "artificial" middle tier for security, where users are only able to use the application itself (e.g. they cannot install or use MSSQL clients, or "hacking" tools). Care should be taken implementing this approach by disabling code execution possibilities in the terminal server environment (application whitelisting) and configuring strict software restriction policies and file permissions.

We recommend careful testing before applying any of the recommended solutions.

References

N/A

BIN-2: Unsafe Object Deserialization

Targets	Finding Status	Overall Risk	High
Sample Application v1.0	Open	Impact	High
		Likelihood	Medium

Description

We found that the `Sample Application` applications used object deserialization. The client communicated with the server by sending and receiving serialized objects. These objects were deserialized without any checks before the execution of their designated method.

The insecure deserialization was performed by the `OpenMessage method defined in the 'Sample.Messages' class. The following command was used to generate a payload that was received by the application and deserialized. After deserialization the ping command was executed by the application:

```
d:\void\tools\ysoserial.net>ysoserial.exe -f BinaryFormatter -g TypeConfuseDelegate -o
base64 -c "ping collaboratorserver"
```

Impact

Deserializing uncontrolled objects from an untrusted source such as a remote client can lead to unexpected security consequences and vulnerabilities because their content is uncertain. This might lead to malicious code execution.

Likelihood

An attacker with a valid user account might be able to send a message to the victim user that contains malicious code.

Recommendation

We recommend using data structures (e.g., JSON) for data exchange and function calls instead of serialization.

Microsoft suggests applications should stop using BinaryFormatter serialization as it is considered insecure. For more information, please see: https://learn.microsoft.com/en-us/dotnet/standard/serialization/binaryformatter-security-guide

References

N/A

BIN-3: Lack of Code Obfuscation

Targets	Finding Status	Overall Risk	Low
Sample Application v1.0	Open	Impact	Low
		Likelihood	Medium

Description

We found that the `Sample Application` application's main components were using C#, and no obfuscation was used on them.

Due to the nature of .NET language, the compiled binaries may contain a large amount of metadata (Common Intermediate Language), which makes it possible to decompile the .NET assemblies to a readable source code resembling the original.

For example, below is the partially decompiled source code of the `OpenMessage` and 'DeserializeMessage' functions of the found in SampleApplication.exe file:

```
***TRUNCATED***
    private object DeserializeMessage(byte[] data)
    {
        using (MemoryStream memoryStream = new MemoryStream(data))
        {
            BinaryFormatter binaryFormatter = new BinaryFormatter();
            return binaryFormatter.Deserialize(memoryStream);
        }
    }
    private void OpenMessage(object message)
    {
        Logmessage("Message received:");
        WriteMessage(message.ToString());
    }
***TRUNCATED***
```

Impact

An adversary having access to the client binaries might be able to decompile the application to gain access to 'easy-to-understand' source code. Analyzing the source code might aid the attacker in understanding the 'Sample Application' application and its components, that might be used to plan attacks.

Likelihood

An attacker needs to have access to the binaries to decompile them.

- In order to decompile the application, moderate technical skills are sufficient.
- In order to conduct further research and identify possible vulnerabilities in the decompiled source code, advanced technical skills might be required.

Recommendation

We recommend using obfuscation on the 'Sample Application' application and its modules in production environment, after careful testing.

It is necessary to emphasize that obfuscation does not completely prevent reverse engineering attempts, however it might avert less determined attackers from analyzing the application.

References

N/A

Testing Infrastructure and Tool Details

Description of the testing infrastructure:

The below table contains the details of the operating systems used during the assignment:

Table 6 - Details of the operating systems used during the assignment

Operating System	Used Version
Kali Linux	Rolling 2023.1
MacOS Ventura	Version 13.1

We performed our external facing test using the following public IP addresses:

Table 7 - Details of the public IP address used during testing

IPv4 Address	ISP	ASN	Country	City
81.183.236.252	Magyar Telekom	AS5483	Hungary	Budapest

List of Tools used during testing:

The below table gives an overview of the specific tools we have used during this specific assignment.

Note: Generic operating system utilities such GNU packages within a Linux distribution will not be listed below, furthermore if a tool has been updated or upgraded during the testing period, only the latest used version information will be listed below.

Table 8 - Details of the tools used during the assessment

Tools Name	Description	Used Version
Tool #1	Tool Description #1	Version 1.0.
Tool #2	Tool Description #2	N/A

Finding Risk Details

Risk Details

We use the following risk ratings: Informational, Low, Medium, High and Critical. These categories offer limited accuracy however they represent that during an assignment the complete context and relevant technical information related to findings may only be partially available.

During the risk calculation process due to the limited information, time and resources of an assignment and specific goal, we do not “compress” or “inflate” the calculated findings to be accurate on an organizational level. The calculated risk does not represent a potential priority for mitigation on an organizational level, as other risks may be present in out-of-scope systems.

The Informational category is reserved for findings that are by themselves do not represent direct risk to Sample Company, however the finding’s existence may hold valuable information.

Given that new vulnerabilities in operating systems and applications are discovered daily and system configurations change frequently, these results are based on publicly available vulnerability information and the system configurations at the time of testing. Consequently, new vulnerabilities may have emerged since then, or existing weaknesses may have been addressed.

Customers should perform their own assessments to align these findings with their information security management risk rating methodology. Additionally, customers should determine if and how to implement the recommended remediation activities. A thorough risk rating requires internal knowledge of the company's business requirements and organizational processes.

Note: Please note that the report may include "False Positives" due to a lack of access to all necessary system or design information. Therefore, all results must be verified to confirm their corresponding risk.

The following Risk Calculation Matrix describes how the risk rating and potential advice for further action is derived from the Likelihood and Impact:

Impact \ Likelihood				
High	Medium	High	Critical
Medium	Low	Medium	High
Low	Low	Low	Medium
Informational	Low	Medium	High

Table 9 – Risk Calculation Matrix

Impact Details

Quantifying potential damage without a detailed financial risk assessment is challenging, so we use a scenario-based approach. To provide a comprehensive overview of the risk environment in the audited areas, we consider various factors, including but not limited to the confidentiality, integrity, and availability of information; the impact on technical operations; the potential effects on reputation and customer satisfaction; the risk of potential penalties or legal consequences; the financial impact; the implications for customer safety; and other relevant aspects.

Our professional judgement, based on our experience, leads us to categorize vulnerabilities by their potential impact.

High: Findings that enable attackers to manipulate sensitive information, damage Sample Company's reputation, obtain unauthorized access to sensitive information (such as financial information or Personal Identifiable Information), or obtain unauthorized access to the applications or infrastructure of Sample Company are to be categorized as high impact.

Medium: Medium impact findings are that may still allow attackers to damage Sample Company's reputation or obtain unauthorized access to Sample Company's information, however the privileges an attacker could obtain by taking advantage of these vulnerabilities are limited and are likely to depend on other weaknesses and other resources to be usable for accessing sensitive information.

Low: Low impact findings do not pose an immediate threat, but may ultimately expose sensitive information, offering valuable intelligence to potential attackers. These vulnerabilities provide only restricted access to the system and limited operational risk to Sample Company.

Note: Please note that in the described system, impact by itself cannot be classified as critical.

Likelihood Details

The likelihood of a vulnerability being exploited is assessed based on three critical factors: the ease of discovery, the number of potential attackers with access to it, and the resources and expertise needed to execute an exploit. The probability of exploitation is then categorized as follows:

High: Findings with high likelihood can be easily identified and exploited by malicious actors. They are broadly accessible to numerous attackers, requiring only minimal technical expertise and resources to leverage.

Medium: Medium-likelihood findings require a more advanced approach for exploitation. They are not easily detectable and are accessible to a limited pool of attackers. Exploiting these vulnerabilities demands higher technical expertise or significant resources.

Low: Low-likelihood vulnerabilities are challenging to exploit due to their limited discoverability, restricted access to a small number of potential attackers, or the need for advanced expertise and sophisticated resources to execute an exploit.

Note: Please note that in the described system, likelihood by itself cannot be classified as critical.