BUGFYND · APPLICATION SECURITY

Thick Client Application Penetration Testing Report

Prepared for: Sample Client Pvt Ltd
Application: DVTA (Damn Vulnerable Thick Client App)

Version 1.1 · Final report

SAMPLE REPORT: findings come from a deliberately vulnerable lab application, not from a real client

Confidentiality statement

This document is confidential and intended only for the client named on the cover. It may not be copied, shared or published, in whole or in part, without written permission from the client and Bugfynd. The client may share it with its own auditors, customers and partners to support its security and compliance programs.

About this sample: this report shows the format and level of detail Bugfynd delivers after a thick client penetration test. All findings come from DVTA, an open-source, intentionally vulnerable training application tested in our lab. Client, dates and evidence are illustrative; secrets are redacted.

Contents

#	Section
1	Document details
2	Introduction, scope and duration
3	Executive summary and risk ratings
4	Approach and methodology
5	Results overview and high-level recommendations
6	Detailed findings (6)
7	Conclusion
A–C	Annexures: changes to environment, tools used, tests performed

SAMPLE

1. Document details

Document control

Document title	Thick Client Penetration Testing Report
Classification	Confidential – Final report
Report version	1.1 (includes retest results)

Document history

Version	Date	Prepared by	Status
1.0	[Date]	Bugfynd testing team	Initial report
1.1	[Date]	Bugfynd testing team	Final report after retest

Client information

Company	Sample Client Pvt Ltd
Point of contact	[Name, title]
Email	[contact@client.example]

Bugfynd team

Name	Role	Responsibility
Consultant A	Lead Consultant	Testing and report preparation
Consultant B	Security Consultant	Technical review
Consultant C	Security Consultant	Testing support and retest
[Client contact]	Client representative	Report acceptance

SAMPLE

2. Introduction

Bugfynd performed a manual penetration test of the client's Windows desktop application to find security weaknesses an attacker could exploit, measure their business impact and give clear steps to fix them. Testing was carried out under written authorization from the client and within the agreed rules of engagement.

Scope

Application	Type	Version	Components
DVTA	.NET Windows desktop (two-tier)	[x.y]	DVTA.exe, configuration files, backend database, FTP backup service

Duration

Activity	Dates
Initial assessment including reporting	[Start date] – [End date]
Retest and report update	[Retest date]

Test approach

- Grey-box testing with standard user and administrator test accounts provided by the client.
- Testing performed from Bugfynd's lab environment on a dedicated Windows test machine.
- No denial-of-service or destructive testing was performed.

Limitations

- Results reflect the application version and environment available during the test window.
- Server-side components were tested only where reachable from the client application.

SAMPLE

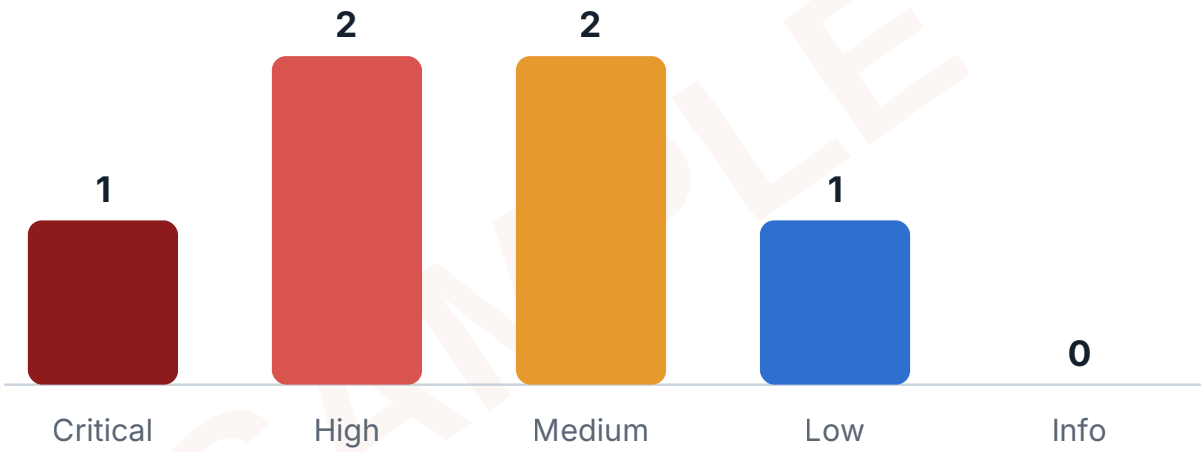
3. Executive summary

Bugfynd identified **6 security findings** in the application: **1 Critical, 2 High, 2 Medium and 1 Low**. The most serious issue lets an attacker log in as an administrator without a password through SQL injection on the login screen. Database credentials stored in the client and cleartext FTP traffic also expose data to local and network attackers.

After the client's fixes, a retest confirmed that **4 of 6** findings are remediated. Two findings (cleartext FTP and sensitive data in memory) remain open with an agreed remediation plan.

1	2	2	1	0
Critical	High	Medium	Low	Info

Findings by severity



SAMPLE

Initial test vs retest

Status	Critical	High	Medium	Low	Total
Initial assessment	1	2	2	1	6
Open after retest	0	1	1	0	2

Risk rating

Each finding is scored with **CVSS v3.1** (Common Vulnerability Scoring System), the industry standard for rating how severe a vulnerability is. Scores run from 0 to 10.

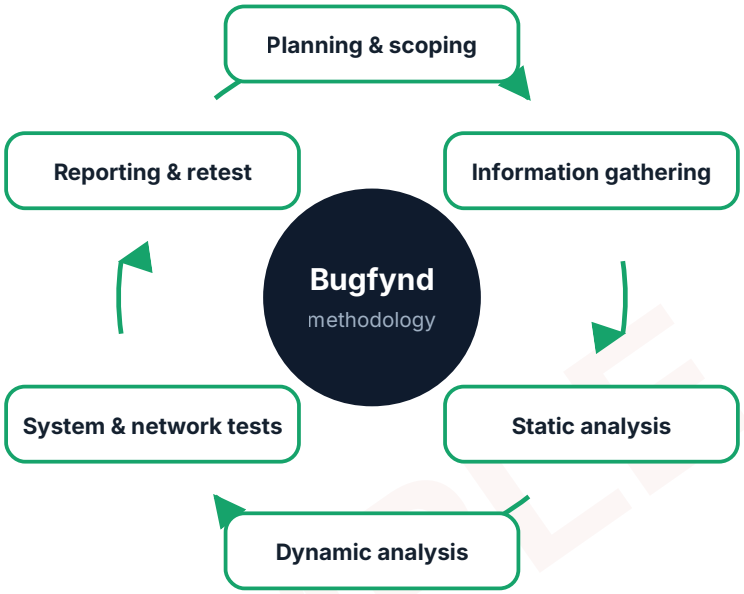
Severity	CVSS score	What it usually means
Critical	9.0 – 10.0	Easy to exploit, often remotely and without credentials; can fully compromise the application or its data. Fix immediately.
High	7.0 – 8.9	Exploitable with little effort; gives access to sensitive data or higher privileges. Fix as a priority.
Medium	4.0 – 6.9	Needs specific conditions or some access; limited impact on its own but can be chained. Fix in the next release.
Low	0.1 – 3.9	Minor impact or hard to exploit; mainly defence-in-depth. Fix as part of normal hardening.
Info	0.0	Best-practice observation with no direct security impact.

Scores are calculated with the FIRST CVSS 3.1 calculator (first.org/cvss/calculator/3.1) and adjusted for the client's environment where relevant.

SAMPLE

4. Approach and methodology

Bugfynd's thick client methodology combines manual testing with tool support and follows recognised standards: **OWASP Desktop App Security Top 10**, **OWASP Top 10**, **OWASP API Security Top 10**, **CWE Top 25** and the **Penetration Testing Execution Standard (PTES)**.



Phase	What we do
Planning & scoping	Agree scope, test accounts, testing windows and contacts; obtain written authorization.
Information gathering	Map the architecture, technologies, entry points, files, registry use, network endpoints and dependencies.
Static analysis	Decompile binaries (dnSpy, ILSpy), review configuration files and stored data, look for hard-coded secrets and weak cryptography, check binary protections.
Dynamic analysis	Test inputs for injection, authentication and authorization flaws, GUI control bypass, file handling and business logic while the application runs.
System & network tests	Inspect memory, file and registry permissions, DLL loading and traffic (TLS, cleartext protocols, request tampering).
Reporting & retest	Document every finding with impact, steps, evidence and fixes; report critical issues the same day; retest after fixes.

Communication

- Daily status updates during testing.
- Critical findings reported to the client contact the same day.
- Walkthrough call with the development team after report delivery.

SAMPLE

5. Results overview

#	Finding	Severity	CVSS	Retest status
1	SQL Injection in Login Function	Critical	9.8	Remediated
2	Hard-coded Database Credentials in Configuration	High	7.8	Remediated
3	Cleartext Credentials Sent over FTP	High	7.4	Open
4	Sensitive Data Retained in Process Memory	Medium	5.5	Open
5	DLL Hijacking via Writable Search Path	Medium	6.7	Remediated
6	Missing Binary Protections (ASLR / DEP / CFG)	Low	3.3	Remediated

High-level recommendations

1. Move all authentication and database access to a server-side API; use parameterized queries everywhere.
2. Remove credentials and secrets from the client; use a secrets vault or Windows DPAPI where local secrets are unavoidable.
3. Replace FTP with an encrypted channel (SFTP or HTTPS) and enforce TLS 1.2 or higher.
4. Clear sensitive data from memory after use.
5. Harden the installation: safe DLL loading, restricted folder permissions, signed binaries and modern compiler protections.

SAMPLE

6.1 SQL Injection in Login Function

Critical

Severity / CVSS	Critical · 9.8
CVSS vector	`CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H`
Affected component	DVTA.exe – Login form
Category	CWE-89
Retest status	Remediated
Description	The username and password fields on the login screen are inserted directly into a SQL query sent to the backend database. A crafted value in the username field changes the query logic and lets an attacker log in without valid credentials.
Impact	An attacker can bypass authentication, log in as any user including the administrator, and read or change data in the backend database.
Steps to reproduce	1. Launch DVTA.exe and open the login screen. 2. Enter `' OR '1'='1' --` as the username and any value as the password. 3. Click Login: the application signs in as the first user returned by the database (admin).

Evidence	`Username: ' OR '1'='1' --`
	`Password: test`
	`Result  : Logged in as 'admin' (Administrator panel shown)`
	Screenshots and full request/response logs are attached in the evidence pack.

Recommendation	• Use parameterized queries or stored procedures for all database access. • Validate input length and characters on both the client and the server. • Move authentication to a server-side API so the client never builds SQL.
Reference	https://owasp.org/www-community/attacks/SQL_Injection

SAMPLE

6.2 Hard-coded Database Credentials in Configuration

High

Severity / CVSS	High · 7.8
CVSS vector	`CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:H/A:H`
Affected component	DVTA.exe.config
Category	CWE-798
Retest status	Remediated
Description	The application configuration file stores the database connection string, including the database username and password. The value is only lightly encrypted with a key that is also present in the decompiled code, so it can be recovered in minutes.
Impact	Any user with access to the installation folder can recover database credentials and connect directly to the database, bypassing all application controls.
Steps to reproduce	1. Open the installation folder and view DVTA.exe.config. 2. Decompile DVTA.exe with dnSpy and locate the decryption routine and key. 3. Use the key to decrypt the stored connection string.

Evidence	`<add key="DBPASSWORD" value="[REDACTED – encrypted value]"/>`
	`Decryption key found in: DVTA.Login.decryptPassword()`
	`Recovered: user=sa  password=[REDACTED]`
	Screenshots and full request/response logs are attached in the evidence pack.

Recommendation	• Never store database credentials in the client. • Route all data access through an authenticated server-side API. • If secrets must exist on the device, protect them with Windows DPAPI or a secrets vault.
Reference	https://cwe.mitre.org/data/definitions/798.html

SAMPLE

6.3 Cleartext Credentials Sent over FTP

High

Severity / CVSS	High · 7.4
CVSS vector	`CVSS:3.1/AV:A/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:N`
Affected component	DVTA.exe – Backup to FTP
Category	CWE-319
Retest status	Open
Description	The 'Backup Data' feature uploads files to an FTP server. FTP does not encrypt traffic, so the FTP username, password and uploaded data are visible to anyone on the same network.
Impact	An attacker on the same network can capture FTP credentials and backup files, leading to data theft and unauthorized access to the FTP server.
Steps to reproduce	1. Start Wireshark on the test machine. 2. Log in to DVTA as a normal user and click 'Backup Data to FTP'. 3. Filter the capture with `ftp` and read the USER and PASS commands.

Evidence	`Request: USER dvta`
	`Request: PASS [REDACTED – sent in cleartext]`
	`Response: 230 Login successful.`
	Screenshots and full request/response logs are attached in the evidence pack.

Recommendation	• Replace FTP with SFTP or HTTPS uploads. • Use per-user, short-lived credentials or tokens for uploads. • Restrict the backup server to known client IP ranges.
Reference	https://cwe.mitre.org/data/definitions/319.html

SAMPLE

6.4 Sensitive Data Retained in Process Memory

Medium

Severity / CVSS	Medium · 5.5
CVSS vector	`CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:H/I:N/A:N`
Affected component	DVTA.exe – running process
Category	CWE-316
Retest status	Open
Description	After login, the user's password and the decrypted database connection string remain in process memory as plain strings for the whole session.
Impact	Malware or a local attacker who can read process memory or a crash dump can recover passwords and database credentials.
Steps to reproduce	1. Log in to DVTA. 2. Create a memory dump of DVTA.exe with Process Hacker. 3. Search the dump for the password and for 'Data Source='.

Evidence	`Search 'Data Source=' in DVTA.dmp -> 1 match`
	`Search '<user password>' in DVTA.dmp -> 3 matches`
	Screenshots and full request/response logs are attached in the evidence pack.

Recommendation	• Clear sensitive values from memory as soon as they are no longer needed. • Avoid holding secrets in immutable strings; use SecureString or byte arrays that can be wiped. • Remove database credentials from the client entirely (see finding 2).
Reference	https://cwe.mitre.org/data/definitions/316.html

SAMPLE

6.5 DLL Hijacking via Writable Search Path

Medium

Severity / CVSS	Medium · 6.7
CVSS vector	`CVSS:3.1/AV:L/AC:L/PR:H/UI:N/S:U/C:H/I:H/A:H`
Affected component	DVTA installation folder
Category	CWE-427
Retest status	Remediated
Description	The application tries to load a DLL that does not exist from its own folder before searching system locations. A local user can place a malicious DLL with that name in the folder, and it runs when the application starts.
Impact	An attacker with write access to the folder can run code in the context of the user who starts the application, which can lead to persistence or privilege escalation.
Steps to reproduce	1. Run Procmon with filters Process Name = DVTA.exe and Result = NAME NOT FOUND. 2. Note the DLL loaded from the application folder that is missing. 3. Place a harmless test DLL with that name in the folder and start DVTA: the test DLL is loaded.

Evidence	`DVTA.exe  CreateFile  C:\DVTA\[missing].dll  NAME NOT FOUND`
	`Test DLL loaded on next start: message box displayed`
	Screenshots and full request/response logs are attached in the evidence pack.

Recommendation	• Load libraries with full paths or SetDefaultDllDirectories / LOAD_LIBRARY_SEARCH_SYSTEM32. • Install the application under Program Files with standard user write access removed. • Sign all application binaries and verify signatures before loading.
Reference	https://attack.mitre.org/techniques/T1574/001/

SAMPLE

6.6 Missing Binary Protections (ASLR / DEP / CFG)

Low

Severity / CVSS	Low · 3.3
CVSS vector	`CVSS:3.1/AV:L/AC:L/PR:L/UI:N/S:U/C:N/I:L/A:N`
Affected component	Bundled native DLLs
Category	CWE-693
Retest status	Remediated
Description	Some bundled native libraries are compiled without modern exploit mitigations such as ASLR, DEP and Control Flow Guard.
Impact	If a memory corruption bug exists, missing mitigations make it much easier to turn into reliable code execution.
Steps to reproduce	1. Run PESecurity against all .exe and .dll files in the installation folder. 2. Review the ASLR, DEP and CFG columns.

Evidence	`Module  ASLR  DEP  CFG`
	`[helper].dll  False  False  False`
	Screenshots and full request/response logs are attached in the evidence pack.

Recommendation	• Rebuild native components with /DYNAMICBASE, /NXCOMPAT and /guard:cf. • Add a binary-hardening check to the build pipeline.
Reference	https://learn.microsoft.com/en-us/cpp/build/reference/guard-enable-control-flow-guard

SAMPLE

7. Conclusion

The application had serious weaknesses in authentication, secret handling and data in transit. The most urgent risk, SQL injection on the login screen, has been fixed and verified, along with hard-coded credentials, DLL hijacking and missing binary protections. Bugfynd recommends fixing the two remaining findings before the next release and repeating the test after major changes or at least once a year.

Annexure A: Changes to environment

No permanent changes were made to the client environment. Test files placed during DLL hijacking verification were removed at the end of testing.

Annexure B: Tools used

Tool	Purpose
dnSpy / ILSpy	Decompiling and debugging .NET binaries
Sysinternals (Procmon, Process Explorer, AccessChk)	Process, file, registry and permission analysis
Process Hacker	Memory inspection and process dumps
Wireshark	Network traffic capture and analysis
Echo Mirage / Burp Suite	Intercepting and modifying application traffic
PESecurity	Checking binary protections (ASLR, DEP, CFG)
Regshot	Comparing registry changes
sslscan	TLS configuration review

SAMPLE

Annexure C: Tests performed

Test area	Status
Information gathering and architecture review	Tested
Authentication and authorization	Tested
Input validation and injection	Tested
GUI control and object permissions	Tested
File and configuration storage	Tested
Registry storage and permissions	Tested
Hard-coded secrets and cryptography	Tested
Memory analysis	Tested
DLL loading and binary protections	Tested
Network communication and TLS	Tested
Traffic interception and tampering	Tested
Error handling and information disclosure	Tested
Business logic	Tested

SAMPLE