

Every build, Every release, Every application, Secured

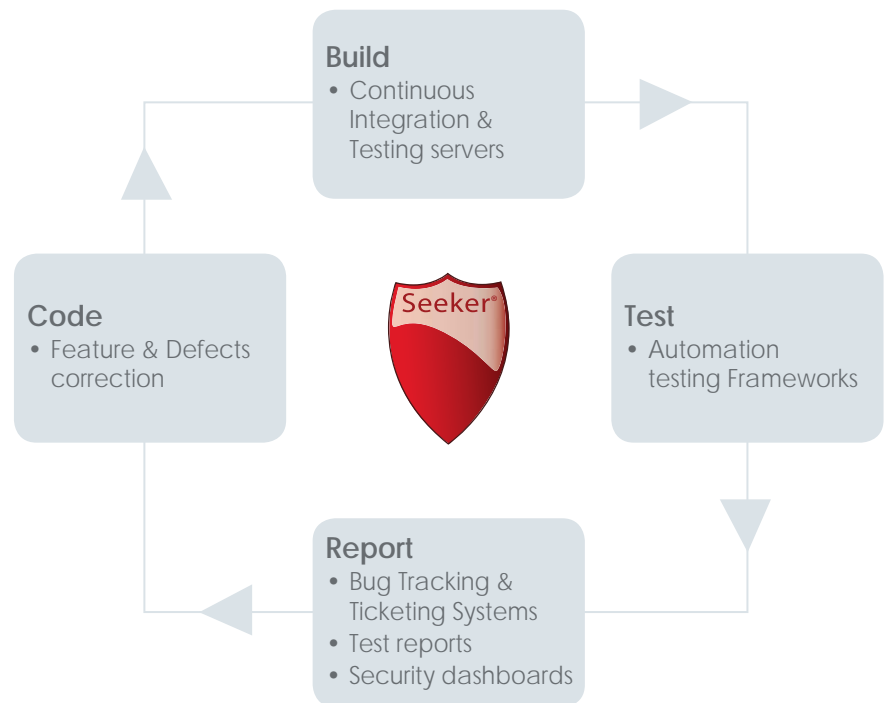
Quotium's Seeker is an Interactive Application Security Testing (IAST) solution which enables organizations to produce secure software efficiently.

It provides very high accuracy in vulnerability detection during the software building and testing process. Seeker uses a unique approach, analyzing code and data flows in runtime to better understand vulnerability context.

Seeker ties vulnerable code to business impact and exploitation, providing a clear explanation of risks. Seeker completely eliminates false positives. This approach makes it possible to confirm or disprove the exploitability and criticality of detected vulnerabilities with no need for human intervention.

Seeker gives a clear view of the security state of applications according to compliance criteria and provides everything needed to secure code and improve security awareness.

Seeker provides a security automation process in the SDLC:



"SEEKER HAS BEEN DESIGNED TO BE FULLY INTEGRATED IN AGILE DEVELOPMENT PROCESS, ALLOWING ALL STAKEHOLDERS TO WORK TOGETHER TOWARDS THE COMMON GOAL OF HACKER-PROOF SOFTWARE"

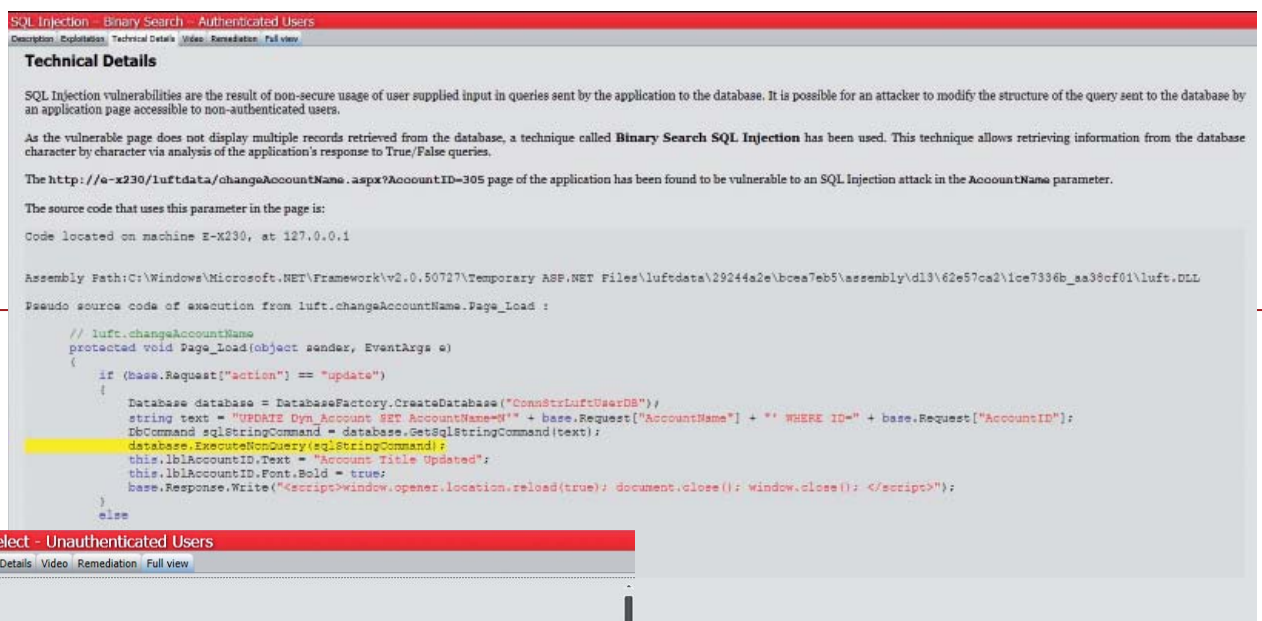
ALL INFORMATION NEEDED TO SECURE THE CODE

The vulnerable lines of code

Seeker provides detailed information regarding the exact location of the vulnerability in the application code, the tier on which the code is deployed, the path the malicious input went through from entering the application to the actual vulnerability.

Detailed context based remediation instructions

Seeker uses information on the application, programming language, framework in use, components and databases to give clear explanations of the problem and the shortest and most effective remediation to fix it. Seeker analysis is carried out without actually requiring the source code itself. This allows security testing of any third party code to integrate in the application, like open source libraries, components developed by external vendors or off-the-shelf products. Seeker can give remediation to secure their interfaces, if their source code is not available.



Technical Details

SQL Injection vulnerabilities are the result of non-secure usage of user supplied input in queries sent by the application to the database. It is possible for an attacker to modify the structure of the query sent to the database by an application page accessible to non-authenticated users.

As the vulnerable page does not display multiple records retrieved from the database, a technique called **Binary Search SQL Injection** has been used. This technique allows retrieving information from the database character by character via analysis of the application's response to True/False queries.

The <http://e-x230/luftdata/changeAccountName.aspx?AccountID=305> page of the application has been found to be vulnerable to an SQL Injection attack in the `AccountName` parameter.

The source code that uses this parameter in the page is:

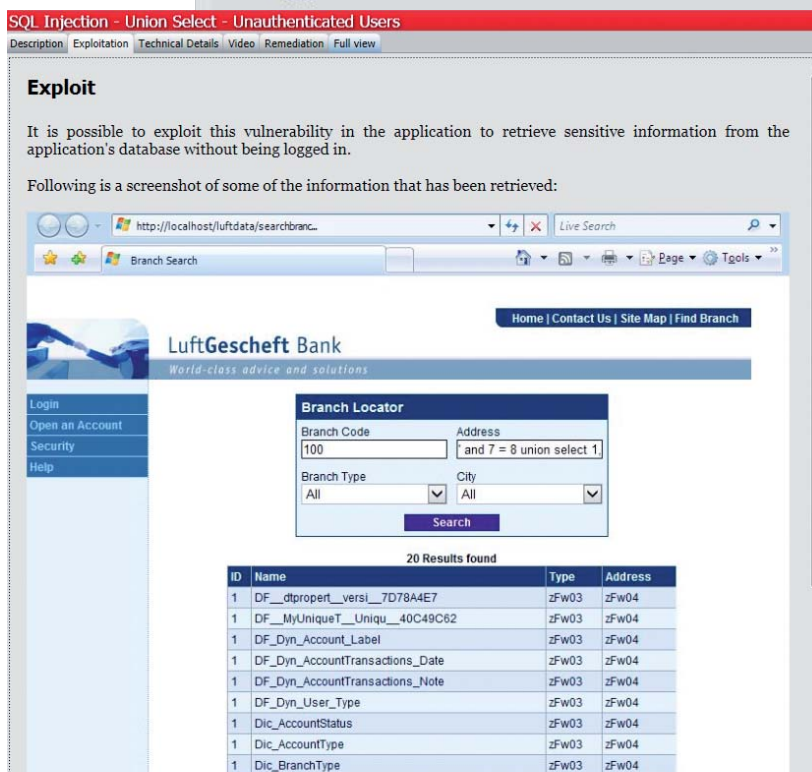
Code located on machine E-X230, at 127.0.0.1

Assembly Path: C:\Windows\Microsoft.NET\Framework\v2.0.50727\Temporary ASP.NET Files\luftdata\29244a2e\bowa7eb5\assembly\d13\62e57ca2\1ce7336b_aa38cf01\luft.DLL

Pseudo source code of execution from `luft.changeAccountName.Page_Load`:

```
// luft.changeAccountName
protected void Page_Load(object sender, EventArgs e)
{
    if (base.Request["action"] == "update")
    {
        Database database = DatabaseFactory.CreateDatabase("ConnStrLuftUserDB");
        string text = "UPDATE Dyn_Account SET AccountName=''" + base.Request["AccountName"] + "' WHERE ID=" + base.Request["AccountID"];
        DbCommand sqlCommand = database.GetSqlCommand(text);
        database.ExecuteNonQuery(sqlStringCommand);
        this.lblAccountID.Text = "Account Title Updated";
        this.lblAccountID.Font.Bold = true;
        base.Response.Write("<script>window.opener.location.reload(true); document.close(); window.close(); </script>");
    }
    else
    {

```

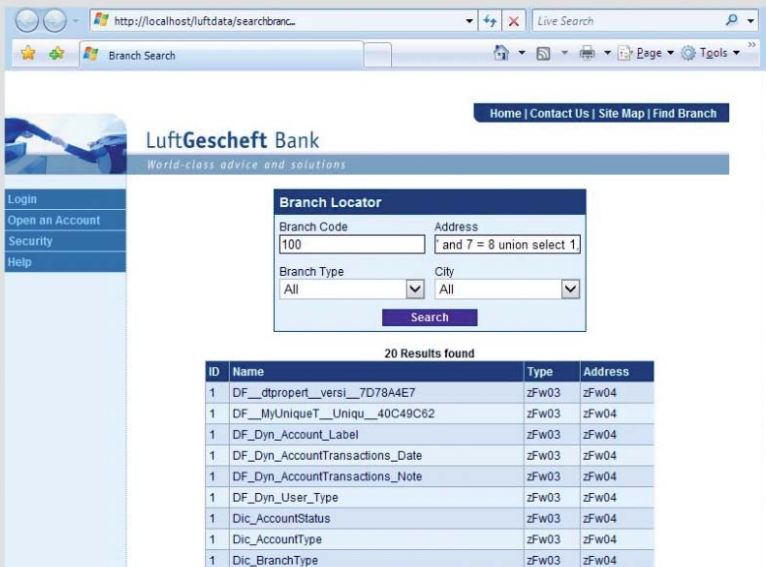


SQL Injection - Union Select - Unauthenticated Users

Exploit

It is possible to exploit this vulnerability in the application to retrieve sensitive information from the application's database without being logged in.

Following is a screenshot of some of the information that has been retrieved:



ID	Name	Type	Address
1	DF__dtpropert__versi__7D78A4E7	zFw03	zFw04
1	DF__MyUniqueT__Uniqu__40C49C62	zFw03	zFw04
1	DF__Dyn_Account_Label	zFw03	zFw04
1	DF__Dyn_AccountTransactions_Date	zFw03	zFw04
1	DF__Dyn_AccountTransactions_Note	zFw03	zFw04
1	DF__Dyn_User_Type	zFw03	zFw04
1	Dic_AccountStatus	zFw03	zFw04
1	Dic_AccountType	zFw03	zFw04
1	Dic_BranchType	zFw03	zFw04

Exploitation explanation

Seeker gives step by step explanations on how the vulnerability can be exploited by a hacker, including videos demonstrating the exploit on the targeted application.

It allows developers, testers and managers to replay the attack and understand the consequences of the vulnerable code.

A CLEAR OVERVIEW OF APPLICATION SECURITY AND COMPLIANCE

Vulnerability reporting

Seeker test reports give an immediate view of the application risk level according to compliance criteria. Compliance can be built per classifications (such as OWASP Top10, SANS/CWE, PCI-DSS...) or according to custom needs.

Seeker Security Management dashboards

The centralized data repository stores information of all projects and tests in the organization. This allows privileged users to monitor and report the overall security risk level, vulnerability trend, compliance status of applications, development teams and projects in the organization.

Seeker allows manager to pinpoint vulnerable systems and teams which need better attention.



SUPPORTED TECHNOLOGIES

→ Operating Systems

Windows XP/2003 or higher (32/64 bit)
Linux (e.g. RedHat, CentOS, Debian, SUSE)
Unix (e.g. Solaris, HP-UX, AIX)

→ Application Technologies

JAVA, .NET, PHP

→ Application Servers

.NET – IIS
Java – Tomcat, WebSphere, WebLogic, Glassfish, JBoss or any J2EE Server
PHP – Apache/IIS

→ Databases

Oracle, MS SQL Server, MySQL, PostgreSQL, DB2

→ Stored Procedures

PL-SQL (Oracle), T-SQL (SQL Server)

INTERFACES

Multiple tools and utilities built into Seeker allow for seamless integration with existing tools and software development processes.

→ Automatic Testing Frameworks

Selenium, IBM Rational Functional Tester, or any other

→ Continuous Integration and Testing servers

Microsoft TFS, HP Quality Center, Hudson, Jenkins, and any other platform via Seeker Command Line activation

→ Bug Tracking and Ticketing Systems

IBM Clear Quest, HP QC, Jira, Bugzilla, Microsoft TFS and more...

ABOUT QUOTIUM

Quotium develops innovative software solutions that guarantee the security and performance of business critical applications throughout their lifecycle. Quotium pioneered Interactive Application Security Testing (IAST) with its security testing software Seeker.



NEW YORK

575 Madison Avenue, 25th fl. New-York, NY 10022, USA
Tel: 212-935-9760 - Fax: 212-755-6385

LONDON

4 Park Place London SW1A 1LP
Tel: +44 (0) 203 178 3681 - Fax: +44 (0) 207 898 9101

PARIS

84/88 bd de la Mission Marchand 92411 Courbevoie Cedex
Tel: +33 (0)1 49 04 70 00 - Fax: +33 (0)1 49 04 71 66

Contact : sales@quotium.com

Follow us on Twitter [@quotium](https://twitter.com/quotium)

www.quotium.com

Quotium

