Methods of information systems protection

UDC 004. 492.3+681.518 **doi:** https://doi.org/10.20998/2522-9052.2022.2.09

Vladyslav Pashynskykh, Yelyzaveta Meleshko, Mykola Yakymenko,
Dmytro Bashchenko, Roman Tkachuk

Central Ukrainian National Technical University, Kropyvnytskyi, Ukraine

RESEARCH OF THE POSSIBILITIES OF THE C# PROGRAMMING LANGUAGE FOR CREATING CYBERSECURITY ANALYSIS SOFTWARE IN COMPUTER NETWORKS AND COMPUTER-INTEGRATED SYSTEMS

Abstract. **The object** of research in the article are the tools and capabilities of the C# programming language for the implementation of cybersecurity analysis software in local computer networks and computer-integrated systems. The relevance of the study is due to the importance of information security of computer and computer-integrated systems in government, military-industrial complex, private business etc., and due to the importance of training cybersecurity professionals in higher education to consider teaching examples in popular programming languages. The **goal** of the work is to research the possibilities of the C# programming language for the development of software that analyzes cybersecurity in local computer networks and computer-integrated systems. The **tasks** to be solved are: *to develop* software for scanning network device ports in computer networks and computer-integrated systems for information security audit, using tools and libraries of the C# programming language, to research the benefits and possibilities of using this programming language for this task. **Research methods:** theory of computer networks, object-oriented programming, theory of algorithms and data structures, theory of software testing. **Conclusions.** In this paper the possibilities and advantages of the C# programming language for developing cybersecurity analysis software for computer and computer-integrated systems were explored. In the course of work software that analyzes information security in local computer networks and computer-integrated systems was developed. This software can be used for educational purposes in learning the C# programming language and cybersecurity of computer systems. The developed software has the potential to be further improved and applied in various fields to test the cybersecurity of local computer networks and computer-integrated technologies.

Keywords: programming language; C#; cybersecurity; vulnerability analysis; port scanner; vulnerability scanner; computer systems; computer-integrated systems; computer networks; information attacks; information security audit.

Introduction

Cybersecurity is important for any activity related to the delimitation of access to information. Government agencies and foreign ministries, the military-industrial complex, private business, including Internet services – this is a small part of the list of consumers interested in keeping their information confidential from third parties.

Computer networks and computer-integrated technologies exist in almost all enterprises. There is a need to analyze their security, because an attacker may appear on the network, who will intercept important information and have a harmful effect on the system, and without analyzing the network, it will be very difficult to find his presence.

The goal of this work is to research of the possibilities of the C# programming language for the development of software that analyzes information security in local computer networks and computer-integrated systems.

The methods of computer network cybersecurity analysis were studied and it was found that one of the basic and mandatory methods of any security analysis is the method based on scanning the ports of devices connected to the local network. It is used both in computer networks [1-3] and in computer-integrated systems, in particular IoT systems [4-6] and smart home systems [7].

In this work, software was created to scan ports in the C# programming language. This uses the Microsoft Visual Studio development environment and Microsoft .NET software technology. C# together with Microsoft .NET software technology allows to most efficiently and quickly develop software for Microsoft Windows operating systems.

The developed application allows to scan the network, get MAC-addresses of devices, scan popular or user-defined ports, store and analyze the received data.

Port scanning is a process that sends client requests to a range of server ports on the host to find the active port, can be used both to test system cybersecurity and to find vulnerabilities with subsequent cyberattack [1]. Administrators often use port scans to check the security policies of their networks. Hackers also use it to identify network services running on the host and search for vulnerabilities. There are two types of port scanning [1-3]:

Availability check. Before starting a comprehensive scan, it will be useful to make sure that the working device is located at the destination address. This task is solved by sending ICMP echo messages using the ping utility with a sequential search of all network IP-addresses. Analyzing traffic and tracking Echo messages sent over a short period of time to all hosts can detect scan attempts.

SYN scan. The port scanner generates IP-packets and monitors responses to them. This technique is called scanning with semi-open connections, because a full TCP/IP session is never fully opened. The port scanner generates a SYN packet. If the port on the destination host is open, the SYN-ACK packet will come from it. In response, the scanner sends an RST packet, which closes the connection before the session is complete.

Port scanners are one of the components of vulnerability scanners [2] – software or hardware for diagnosing and monitoring network computers, which allows to scan networks, computers and programs to identify possible security problems, assess and eliminate vulnerabilities.

The .NET platform and the C# programming language provide all the necessary capabilities to create applications that can interact over a network and use different network protocols [8].

In particular, in the .NET class system, the IP-address is represented by the IPAddress class. Parse() method of this class converts the string representation of the address to IPAddress.

The IPHostEntry class is used to obtain an address on the network. It contains information about a specific host computer. The HostName property returns this hostname, and the AddressList property returns all host IP-addresses, as one computer can have multiple IP-addresses on the network. The Dns class is used to interact with the DNS server and obtain the IP-address. To obtain information about the host computer and its address, it has a method GetHostEntry().

Thus, the C# language contains ready-made tools for working with network devices, which greatly facilitates the creation of software for information security audit of computer networks and computer-integrated systems.

Main material

In this work researching the possibilities of the C# programming language for the development of software that analyzes information security in computer and computer-integrated systems. The purpose of the developing software is checking the server or host for availability on the network, checking its MAC-address and checking open and standard socket ports (port scanning).

The software for cybersecurity analysis of local networks consists of three blocks (Fig. 1), namely: the block of analysis of the local network, the block of analysis of ports of network devices and the block of work with databases and graphical interface.

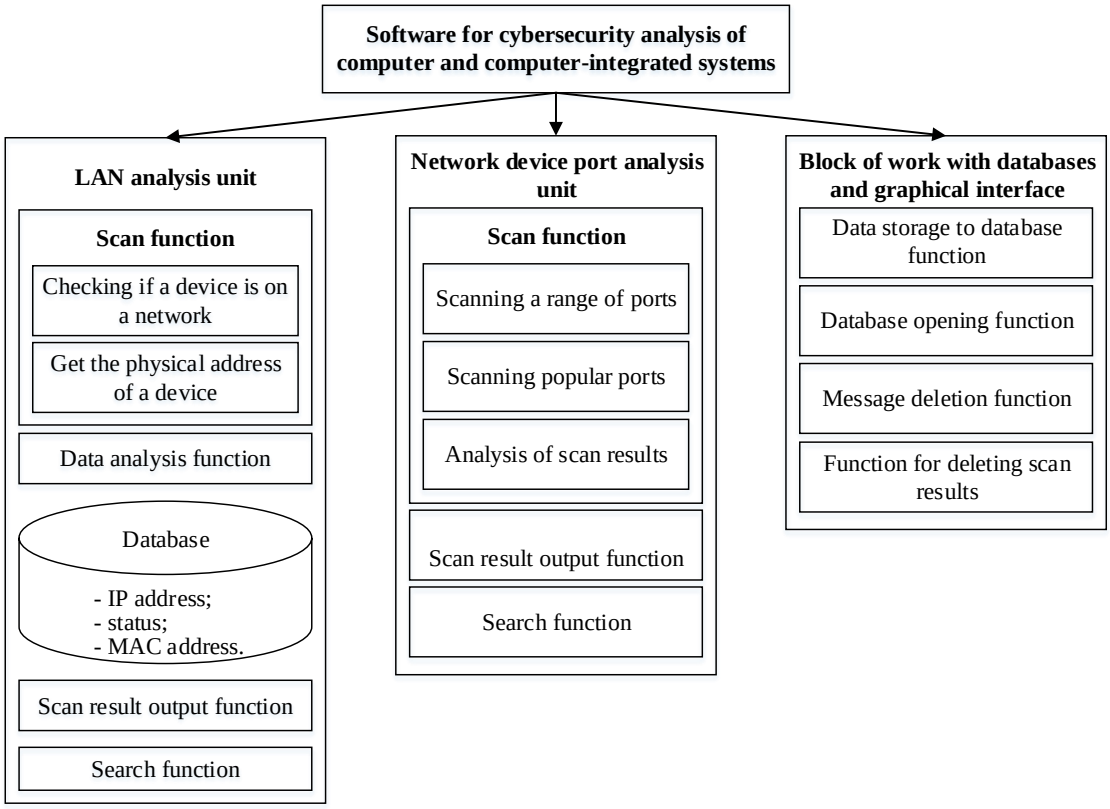

Fig. 1. Structural scheme of the developed software

SQLite is used to work with databases. The peculiarity of SQLite is that it does not use the client-server paradigm, i.e. the SQLite engine is not a separate process with which the application interacts, but provides a library with which the program is compiled and the engine becomes part of the program. Thus, SQLite library function calls (APIs) are used as an exchange protocol. This approach reduces overhead, response time and simplifies the program. SQLite stores the entire database (including definitions, tables, indexes, and data) in a single standard file on the computer on which the application is running. Ease of implementation is achieved due to the fact that before the transaction, the entire file that stores the database is blocked; ACID functions are achieved in particular by creating a log file. Multiple processes or threads can read data from one database at the same time without any problems. An entry in the database can be made only if no other requests are currently being serviced; otherwise the write attempt fails and the error code is returned to the program. Due to the architecture of the SQlite engine, it can be used both on embedded systems and on dedicated machines with gigabyte data sets. A NetDevice class is used to represent the device on the network. Consider him implementation (Fig. 2):

```
// List of devices
List <NetDevice> devList = new List <NetDevice> ();

public class NetDevice {
private string ip;
  private string status;
private string mac;

// Initialization of properties by the constructor
public NetDevice (string [] row) {
ip = row [0];
status = row [1];
mac = row [2];
}

// Obtaining properties
public string [] GetInfo () {
return new string [] {ip, status, mac};
  }
}
```

Fig. 2. Suggested source code of implementing a NetDevice class for representing a device of computer network

Each class object is a device on the network. It has three properties, namely: network address (ip), status, and physical address (mac). Also, each object has a method that allows to get its properties. All objects are stored in the devList. A Port class is used to represent the port. Consider its implementation (Fig. 3):

```
// List of ports
List <Port> portList = new List <Port> ();

public class Port {
private string port;
private string status;
private string description;

// Dictionary with the description of popular ports
Dictionary <int, string> commonports = new Dictionary <int, string> {
[20] = "File Transfer Protocol (FTP) Data Transfer",
[21] = "File Transfer Protocol (FTP) Command Control",
[22] = "Secure Shell (SSH) Secure Login",
[23] = "Telnet remote login service, unencrypted text messages",
[25] = "Simple Mail Transfer Protocol (SMTP) E-mail routing",
[53] = "Domain Name System (DNS) service",
[80] = "Hypertext Transfer Protocol (HTTP) used in the World Wide Web",
[110] = "Post Office Protocol (POP3)",
[119] = "Network News Transfer Protocol (NNTP)",
[123] = "Network Time Protocol (NTP)",
[143] = "Internet Message Access Protocol (IMAP) Management of digital mail",
[161] = "Simple Network Management Protocol (SNMP)",
[194] = "Internet Relay Chat (IRC)",
[443] = "HTTP Secure (HTTPS) HTTP over TLS / SSL"
};

// Initialization of properties by the designer
public Port (string [] row) {
port = row [0];
status = row [1];
if (commonports.ContainsKey (Convert.ToInt32 (row [0])))
description = commonports [Convert.ToInt32 (row [0])];
}
else {
description = "None";
}
}

// Obtaining properties
public string [] GetInfo () {
return new string [] {port, status, description};
}
}
```

Fig. 3. Suggested source code of implementing a Port class to represent the port of a network device

Each this class object is a port. It has three properties, namely: port number, status, and description. The port description is added when the object is initialized automatically. A description of the most popular ports is

stored in the Commonports dictionary. Also, each object has a method that allows to get its properties. All objects are stored in the sorted portList.

As mentioned above, SQLite is used to work with databases. Consider the function where data is saved to the database (Fig. 4).

```
// Save the device list to the * .sqlite database file
private void saveAsToolStripMenuItem_Click (object sender, EventArgs e) {
// Generate a database file name using the current date and time
var time = DateTime.Now;
string formattedTime = time.ToString ("ddMMyyyy_hhmmss");

// Create a save file dialog
SaveFileDialog saveFile = new SaveFileDialog ();
saveFile.Title = "Save database as ...";
saveFile.FileName = "ScanResult_" + formattedTime + ".sqlite";
saveFile.Filter = "SQLite DB files (* .sqlite) | * .sqlite | All files (*. *) | *. *";
//savefile.InitialDirectory = @ "C: \"; // Uncomment to use another default save dir, ie C: \
if (saveFile.ShowDialog () == DialogResult.OK) {
try {
// Create a database file
SQLiteConnection.CreateFile (saveFile.FileName);

// Create a connection
SQLiteConnection m_dbConnection;

// Connect to the database
m_dbConnection = new SQLiteConnection ("Data Source =" + saveFile.FileName + "; Version = 3;");
m_dbConnection.Open ();

// Create a table
string sql;
sql = "create table iplist (ip varchar (20), status varchar (5), mac varchar (30))";
SQLiteCommand command;
command = new SQLiteCommand (sql, m_dbConnection);
command.ExecuteNonQuery ();

// Write device list data to the database
foreach (var item in devList) {
command = new SQLiteCommand (m_dbConnection);
command.CommandText = "insert into iplist (ip, status, mac) values (@ip, @status, @mac)";
command.CommandType = CommandType.Text;
command.Parameters.Add (new SQLiteParameter ("@ ip", item.GetInfo (([0])));
command.Parameters.Add (new SQLiteParameter ("@ status", item.GetInfo (([1])));
command.Parameters.Add (new SQLiteParameter ("@ mac", item.GetInfo (([2])));
command.ExecuteNonQuery ();
}

// Close the database connection
m_dbConnection.Close ();

// Send a message to the user
writeMessage ("\ r \ nDatabase successfully saved.");
}
catch (Exception ex) {
MessageBox.Show (Error: Could not write file to disk. Original error: "+ ex.Message," Error ");
}
}
}
```

Fig. 4. Suggested source code of implementing the function of saving scan results to a SQLite database

The table of the created database contains three columns, namely: “ip”, “status” and “mac”.

Consider the scanning function of the local network, which uses multithreading to increase the scanning speed (Fig. 5).

Multithreading is a property of an operating system or application that means that the process generated in the operating system can consist of several threads running in parallel or even simultaneously on multiprocessor systems. For some tasks, this separation can lead to more efficient use of computer resources. Such execution threads are also called streams.

The essence of multithreading is quasi-multitasking at the level of one executable process, i.e. all threads are executed in the address space of the process. In addition, all process threads have not only a common address space, but also common file descriptors. The process being performed has at least one (main) thread.

To implement multithreading, we use the Parallel.For loop built into Microsoft .NET software technology, which automatically creates the maximum number of threads to speed up certain actions.

The iphlpapi.dll library is used to obtain the device's MAC-address (Fig. 6).

```
   // Scan IP addresses async
   private void scanIP () {
   string ipone = firstAddrOctetOneTextBox.Text + "." + firstAddrOctetTwoTextBox.Text + "." +
firstAddrOctetThreeTextBox.Text + "." + firstAddrOctetFourTextBox.Text;
   string iptwo = secondAddrOctetOneTextBox.Text + "." + secondAddrOctetTwoTextBox.Text + "." +
secondAddrOctetThreeTextBox.Text + "." + secondAddrOctetFourTextBox.Text;
   if (isValidIP (ipone) && isValidIP (iptwo) == true) {
   Invoke (new Action () => {
   // Notify user, that scanning process is starting
   writeMessage ("\ r \ nScanning LAN ...");
   }));

   int start = BitConverter.ToInt32 [new byte]]
   Convert.ToByte (firstAddrOctetFourTextBox.Text),
   Convert.ToByte (firstAddrOctetThreeTextBox.Text),
   Convert.ToByte (firstAddrOctetTwoTextBox.Text),
   Convert.ToByte (firstAddrOctetOneTextBox.Text)
   }, 0);

   int end = BitConverter.ToInt32 [new byte]]
   Convert.ToByte (secondAddrOctetFourTextBox.Text),
   Convert.ToByte (secondAddrOctetThreeTextBox.Text),
   Convert.ToByte (secondAddrOctetTwoTextBox.Text),
   Convert.ToByte (secondAddrOctetOneTextBox.Text)
   }, 0);

   // Create and fill list of IP addresses
   List <IPAddress> addresses = new List <IPAddress> ();
   for (int i = start; i <= end; i ++) {
   byte [] bytes = BitConverter.GetBytes (i);
   addresses.Add (new IPAddress (new [[{bytes [3], bytes [2], bytes [1], bytes [0]}));
   }

   // Multithreading LAN scan
   Parallel.For (0, addresses.Count, new ParallelOptions (), (i, state) => {
   System.Net.NetworkInformation.Ping p = new System.Net.NetworkInformation.Ping ();
   System.Net.NetworkInformation.PingReply rep = p.Send (addresses [i] .ToString (), 100);
   if (rep.Status == System.Net.NetworkInformation.IPStatus.Success) {
   NetDevice device = new NetDevice (new string [[{addresses [i] .ToString (), "online", getMAC
(addresses [i [.ToString ())}));
   devList.Add (device);
   }
   else {
   if (showAllChkBox.Checked == true) {
   // Creating new device
   NetDevice device = new NetDevice (new string [] {addresses [i] .ToString (), "offline", "none"});

   // Adding new device to list
   devList.Add (device);
   }
   }
   });

   Invoke (new Action () => {
   // Fill listView
   fillListView ();

   // Notify user, that scanning process is done
   writeMessage ("Done.");
   }));
   }
   else {
   MessageBox.Show ("Error: Invaild IP address.", "Error");
   }
   }
```

Fig. 5. Suggested source code of the implementation of the LAN scanning function, which uses multithreading to increase the scanning speed

This function by sending an ARP request to the appropriate IP-address allows to get its physical address – MAC-address. MAC Address is a communication protocol designed to convert IP-addresses to MAC-addresses on TCP/IP networks.

Developed application allows to scan the network, obtain MAC-addresses of devices, scan popular or user-defined ports, store and analyze the received data.

Consider in more detail the interface of the developed application. The interface consists of two tabs – "LAN Scan" and "Port Scan", a text box in which user messages, analysis results and user menus are displayed.

The tab "LAN Scan" (Fig. 7) consists of the table, eight input fields for four octets of the initial and final IP-address, respectively, input fields for search, one option (checkbox) and three buttons.

```
// Used this library to get the MAC address
[DllImport ("iphlpapi.dll", ExactSpelling = true)]
public static extern int SendARP (int destIp, int srcIP, byte [] macAddr, ref uint physicalAddrLen);

// Sent ARP to obtain MAC address by IP-address
private string getMAC (string ip) {
IPAddress dst = IPAddress.Parse (ip); // IP-address to scan

byte [] macAddr = new byte [6];
uint macAddrLen = (uint) macAddr.Length;

if (SendARP (BitConverter.ToInt32 (dst.GetAddressBytes (), 0), 0, macAddr, ref macAddrLen)! = 0) {
throw new InvalidOperationException ("SendARP failed.");
}

string [] str = new string [(int) macAddrLen];
for (int i = 0; i <macAddrLen; i ++) {
str [i] = macAddr [i] .ToString ("x2");
}

return string.Join (":", p.);
}
```

Fig. 6. Suggested source code of implementing the function of obtaining the MAC-address of devices using the library iphlpapi.dll

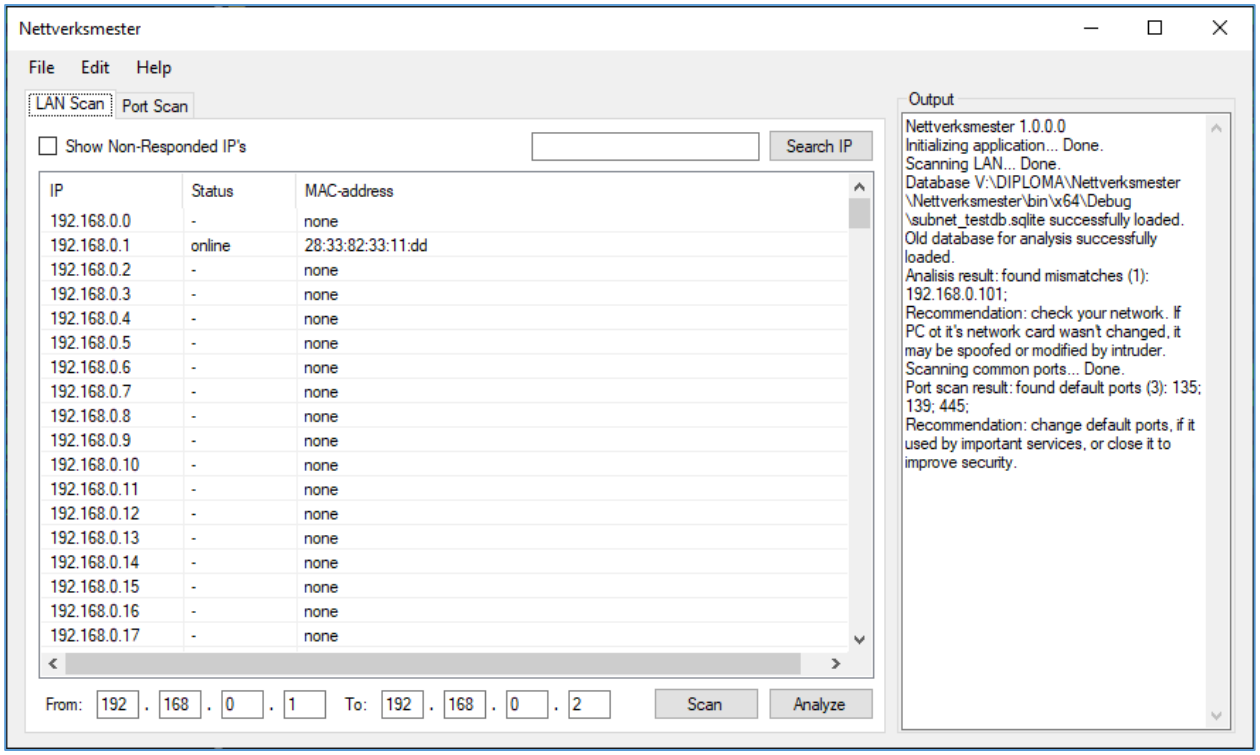

Fig. 7. Application interface. The tab “LAN Scan”

The button “Scan” scans the network and displays the result in the table (Fig. 7).

The result of the scan is a list of IP-addresses with their status (whether the device is located at the address), and MAC device addresses, if available on the network.

The button “Analyze” analyzes the network and reports whether a device has been modified or replaced by an attacker.

The button “Search IP” allows to search for the device by its address.

It will be displayed in the table (Fig. 8). Option “Show Non-Responded IP’s” allows to hide or show IP-addresses that do not belong to any device.

The tab “Scan Port” (Fig. 9) consists of the table, two input fields of the port range for custom scan mode, four input fields for four octets IP-addresses of the device whose ports to be scanned, search input fields, drop-down menu and two buttons.

The drop-down menu (Fig. 10) allows the user to select the scan mode: scan popular ports, or scan a range of ports (all default ports).

The range of ports to be scanned can be set by the user. The button “Scan” scans the ports and displays the result in the table (Fig. 9).

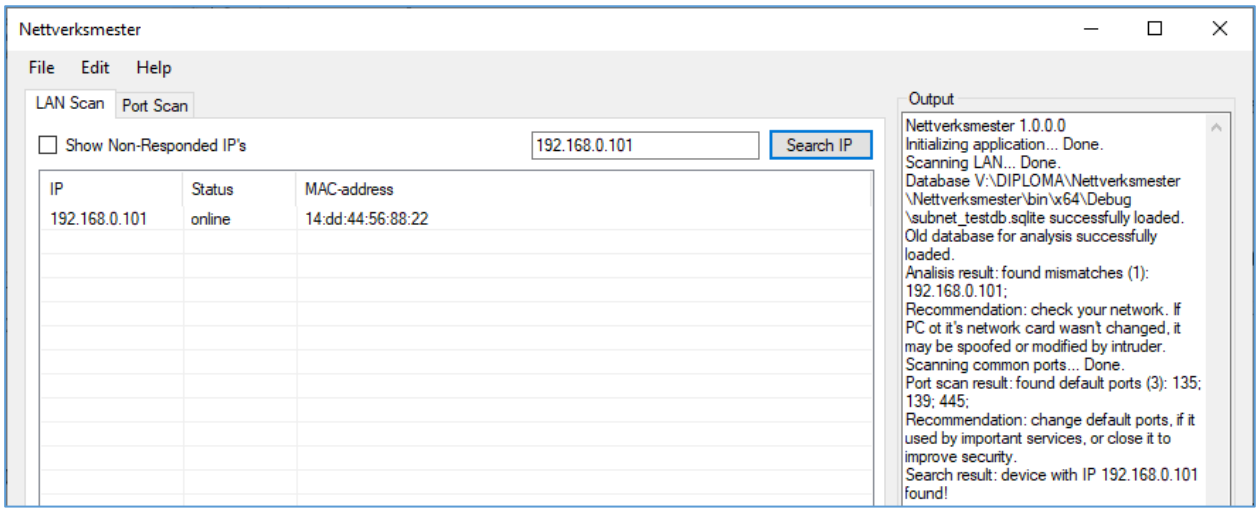

Fig. 8. The tab "LAN Scan". Search for a device on the network and messages

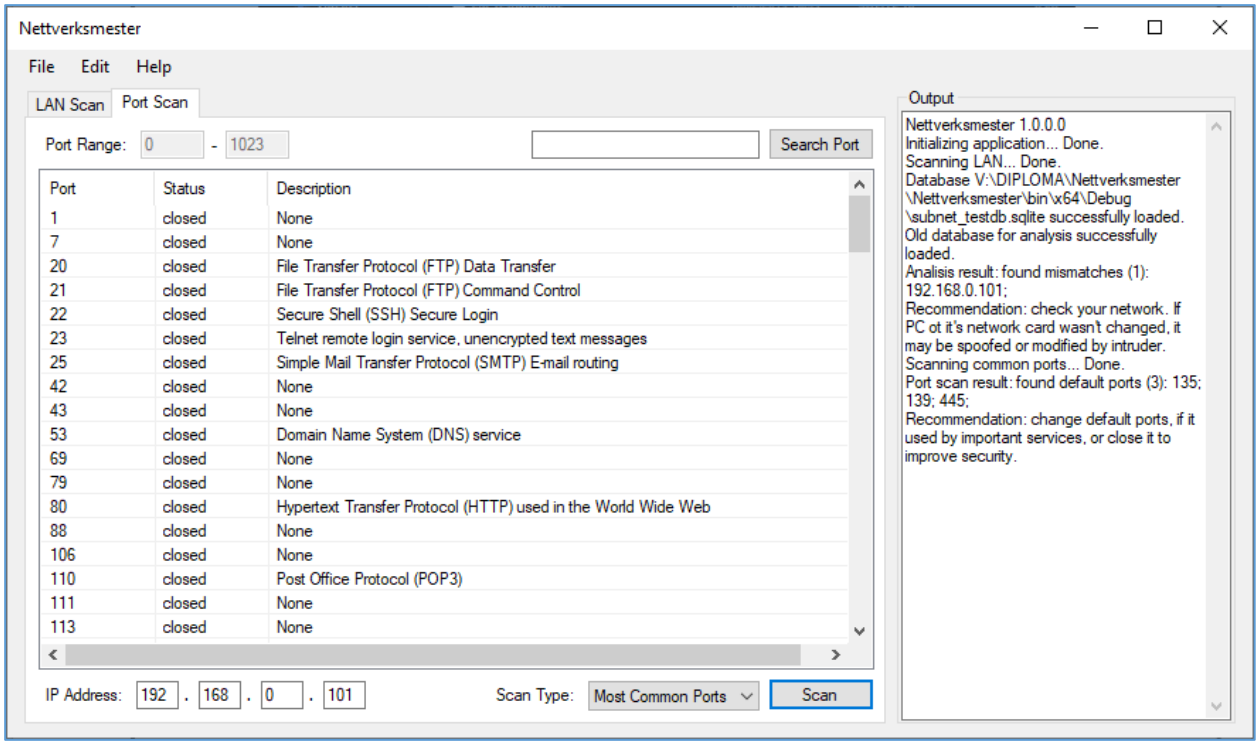

Fig. 9. Application interface. The tab "Port Scan"

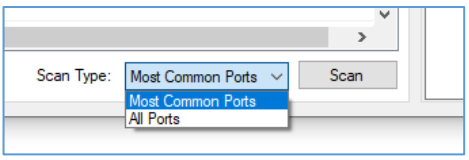

Fig. 10. The tab "Port Scan". Drop-down menu

The result of the scan is a list of ports with their status (open or closed), and a description of the port, if it is popular and used by a particular service.

The button "Search Port" allows to search for a port by its number. It will be displayed in the table (Fig. 11).

Thus by means of the C# programming language managed to create port scanner for cybersecurity audit of computer and computer-integrated systems.

Conclusions

In this paper, the possibilities and advantages of the C# programming language for the development of cybersecurity analysis software in computer and computer-integrated systems were researched.

In the course of work the software which analyzes is developed information security in local computer networks and computer-integrated systems. This software can be used for educational purposes in learning the C# programming language and cybersecurity of computer systems.

The developed software has the potential to be further improved and applied in various fields to test the cybersecurity of local computer networks and computer-integrated technologies.

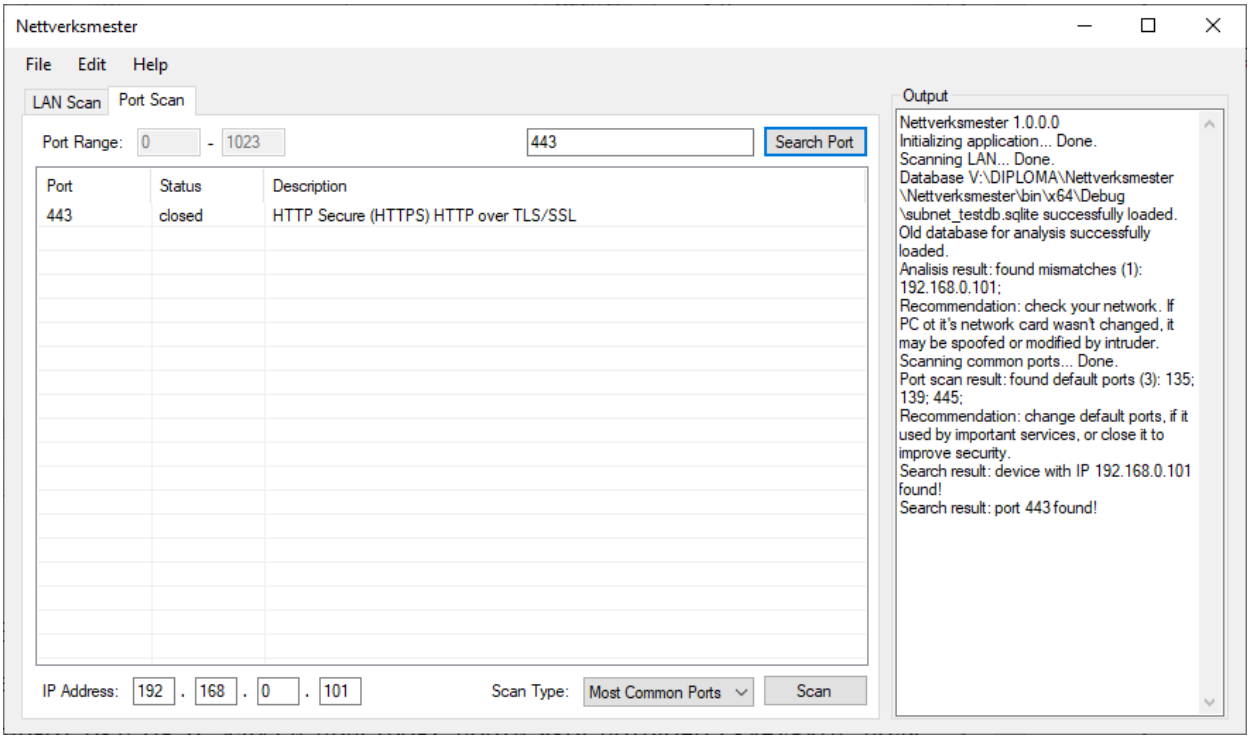

Fig. 11. The tab "Port Scan". Pzzort search and location notification

References

1. Shirey R. (2000), "RFC 2828 Internet Security Glossary", *Internet Engineering Task Force*, available at: https://datatracker.ietf.org/doc/html/rfc2828.
2. Poddubnyi V., Sievierinov O., Pustomelnik O. (2020), "Vulnerability management as an integral part of its security policy", Control, Navigation and Communication Systems. Academic Journal, Vol. 4 (62), Poltava, pp. 55-58, DOI: https://doi.org/10.26906/SUNZ.2020.4.055 (in Ukrainian)
3. Mohammed M. O. (2020), "Automatic Port Scanner", Int.ernational Journal of Innovative Science and Research Technology, Vol. 5(9), pp. 711-717, DOI: https://doi.org/10.38124/IJISRT20SEP503.
4. Verma S., Kawamoto Y., Kato N. (2021) "A Smart Internet-wide Port Scan Approach for Improving IoT Security under Dynamic WLAN Environments", in IEEE Internet of Things Journal, DOI: https://doi.org/10.1109/JIOT.2021.3132389.
5. Verma S., Kawamoto Y., Kato N. (2021) "A Network-Aware Internet-Wide Scan for Security Maximization of IPv6-Enabled WLAN IoT Devices", in IEEE Internet of Things Journal, Vol. 8, No. 10, pp. 8411-8422, DOI: https://doi.org/10.1109/JIOT.2020.3045733.
6. Markowsky L., Markowsky G. (2015) "Scanning for vulnerable devices in the Internet of Things", 2015 IEEE 8th International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), pp. 463-467, DOI: https://doi.org/10.1109/IDAACS.2015.7340779.
7. Bastos D., Shackleton M., El-Moussa F. (2018), "Internet of Things: A Survey of Technologies and Security Risks in Smart Home and City Environments", Living in the Internet of Things: Cybersecurity of the IoT – 2018, DOI: https://doi.org/10.1049/cp.2018.0030.
8. Microsoft (2022), "C# documentation", Official website Microsoft Corporation, available at: https://docs.microsoft.com/en-us/dotnet/csharp.

Received (Надійшла) 21.03.2022
Accepted for publication (Прийнята до друку) 25.05.2022

Відомості про авторів / About the Authors

Пашинських Владислав Вадимович – магістр з комп'ютерної інженерії, кафедра кібербезпеки та програмного забезпечення, Центральноукраїнський національний технічний університет, Кропивницький, Україна;
Vladyslav Pashynskykh – Master of Science in Computer Engineering, Cybersecurity and Software Department, Central Ukrainian National Technical University, Kropyvnytskyi, Ukraine;
e-mail: vlad.pashynskykh@gmail.com; ORCID ID: https://orcid.org/0000-0002-7298-7946.

Мелешко Єлизавета Владиславівна – доктор технічних наук, професор, доцент кафедри кібербезпеки та програмного забезпечення, Центральноукраїнський національний технічний університет, Кропивницький, Україна;
Yelyzaveta Meleshko – Doctor of Engineering Sciences, Professor, Associate Professor of Cybersecurity and Software Department, Central Ukrainian National Technical University, Kropyvnytskyi, Ukraine;
e-mail: elismeleshko@gmail.com; ORCID ID: https://orcid.org/0000-0001-8791-0063.

Якименко Микола Сергійович – кандидат фізико-математичних наук, доцент, завідувач кафедри вищої математики та фізики, Центральноукраїнський національний технічний університет, Кропивницький, Україна;

Mykola Yakymenko – Candidate of Physical and Mathematical Sciences, Associate Professor, Head of Higher Mathematics and Physics Department, Central Ukrainian National Technical University, Kropyvnytskyi, Ukraine;
e-mail: m.yakymenko@gmail.com; ORCID ID: https://orcid.org/0000-0003-3290-6088.

Бащенко Дмитро В'ячеславович – аспірант кафедри кібербезпеки та програмного забезпечення, Центральноукраїнський національний технічний університет, Кропивницький, Україна;
Dmytro Bashchenko – Postgraduate student of Cybersecurity and Software Department, Central Ukrainian National Technical University, Kropyvnytskyi, Ukraine;
e-mail: bashchenko.dv@meta.ua; ORCID ID: https://orcid.org/0000-0003-4561-0016.

Ткачук Роман Олегович – аспірант кафедри кібербезпеки та програмного забезпечення, Центральноукраїнський національний технічний університет, Кропивницький, Україна;
Roman Tkachuk – Postgraduate student of Cybersecurity and Software Department, Central Ukrainian National Technical University, Kropyvnytskyi, Ukraine;
e-mail: tkachuk@outlook.com; ORCID ID: https://orcid.org/0000-0002-1984-0419.

Дослідження можливостей мови програмування C# для створення програмного забезпечення аналізу кібербезпеки в комп'ютерних мережах та комп'ютерно-інтегрованих системах

В. В. Пашинських, Є. В. Мелешко, М. С. Якименко, Д. В. Бащенко, Р. О. Ткачук

Анотація. Об'єктом вивчення у статті є засоби та можливості мови програмування C# для реалізації програмного забезпечення аналізу кібербезпеки у локальних комп'ютерних мережах та комп'ютерно-інтегрованих системах. Актуальність дослідження зумовлена важливістю забезпечення інформаційної безпеки комп'ютерних та комп'ютерно-інтегрованих систем у державних органах, військово-промисловому комплексі, приватному бізнесі тощо, та важливістю під час підготовки фахівців з кібербезпеки у вищих навчальних закладах розгляду навчальних прикладів на популярних мовах програмування. **Метою** роботи є дослідження можливостей мови програмування C# для розробки програмного забезпечення, яке аналізує безпеку у локальних комп'ютерних мережах та комп'ютерно-інтегрованих системах. **Завдання:** розробити програмне забезпечення для сканування портів мережевих пристроїв у комп'ютерних мережах та комп'ютерно-інтегрованих системах для аудиту інформаційної безпеки, використовуючи засоби та бібліотеки мови програмування C#, дослідити переваги та можливості використання саме цієї мови програмування для даної задачі. **Методи досліджень:** теорія комп'ютерних мереж, об'єктно-орієнтоване програмування, теорія алгоритмів та структур даних, теорія тестування програмного забезпечення. **Висновки.** У цій роботі було досліджено можливості та переваги мови програмування C# для розробки програмного забезпечення з аналізу кібербезпеки комп'ютерних та комп'ютерно-інтегрованих систем. У ході роботи розроблено програмне забезпечення, яке аналізує інформаційну безпеку у локальних комп'ютерних мережах та комп'ютерно-інтегрованих системах. Дане програмне забезпечення можна використовувати у навчальних цілях при вивченні мови програмування C# та кібербезпеки комп'ютерних систем. Розроблене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях для перевірки кібербезпеки локальних комп'ютерних мереж та комп'ютерно-інтегрованих технологій.

Ключові слова: мова програмування; C#; кібербезпека; аналіз вразливостей; сканер портів; сканер вразливостей; комп'ютерні системи; комп'ютерно-інтегровані системи; комп'ютерні мережі; інформаційні атаки; аудит інформаційної безпеки.

Исследование возможностей языка программирования C# для создания программного обеспечения анализа кибербезопасности в компьютерных сетях и компьютерно-интегрированных системах

В. В. Пашинских, Е. В. Мелешко, Н. С. Якименко, Д. В. Бащенко, Р. О. Ткачук

Аннотация. Объектом изучения статьи являются средства и возможности языка программирования C# для реализации программного обеспечения анализа кибербезопасности в локальных компьютерных сетях и компьютерно-интегрированных системах. **Актуальность исследования** обусловлена важностью обеспечения информационной безопасности компьютерных и компьютерно-интегрированных систем в государственных органах, военно-промышленном комплексе, частном бизнесе и т.д., и важностью при подготовке специалистов по кибербезопасности в высших учебных заведениях рассмотрения учебных примеров на популярных языках программирования. **Целью работы** является исследование возможностей языка программирования C# для разработки программного обеспечения, анализирующего безопасность в локальных компьютерных сетях и компьютерно-интегрированных системах. **Задача:** разработать программное обеспечение для сканирования портов сетевых устройств в компьютерных сетях и компьютерно-интегрированных системах для аудита информационной безопасности, используя средства и библиотеки языка программирования C#, исследовать преимущества и возможности использования этого языка программирования для данной задачи. **Методы исследований:** теория компьютерных сетей, объектно-ориентированное программирование, теория алгоритмов и структур данных, теория тестирования программного обеспечения. **Выводы.** В этой работе были исследованы возможности и преимущества языка программирования C# для разработки программного обеспечения анализа кибербезопасности компьютерных и компьютерно-интегрированных систем. В ходе работы разработано программное обеспечение, анализирующее информационную безопасность в локальных компьютерных сетях и компьютерно-интегрированных системах. Данное программное обеспечение можно использовать в обучающих целях при изучении языка программирования C# и кибербезопасности компьютерных систем. Разработанное программное обеспечение имеет потенциальную возможность для дальнейшего совершенствования и применения в различных отраслях для проверки кибербезопасности локальных сетей и компьютерно-интегрированных технологий.

Ключевые слова: язык программирования; C#; кибербезопасность; анализ уязвимостей; сканер портов; сканер уязвимостей; компьютерные системы; компьютерно-интегрированные системы; компьютерные сети; информационные атаки; аудит информационной сохранности.