computation

Article

Attack Detection for Healthcare Monitoring Systems Using Mechanical Learning in Virtual Private Networks over Optical Transport Layer Architecture

Vasiliki Liagkou [1], Vasileios Kavvadas [1], Spyridon K. Chronopoulos [1,2,*], Dionysios Tafiadis [3,4], Vasilis Christofilakis [2,*] and Kostas P. Peppas [5]

1 Department of Informatics and Telecommunications, University of Ioannina, GR-47100 Arta, Greece; liagkou@cti.gr (V.L.); vasilis.kavvadas1992@gmail.com (V.K.)
2 Electronics-Telecommunications and Applications Laboratory, Physics Department, University of Ioannina, GR-45110 Ioannina, Greece
3 Department of Speech & Language Therapy, University of Ioannina, GR-45500 Ioannina, Greece; d.tafiadis@ioa.teiep.gr
4 Department of Speech & Language Therapy, Faculty of Health Sciences, European University, 1516 Nicosia, P.O. Box 22006, Cyprus
5 Department of Informatics and Telecommunications, University of Peloponnese, GR-22131 Tripoli, Greece; peppas@uop.gr
* Correspondence: schrono@cc.uoi.gr (S.K.C.); vachrist@uoi.gr; (V.C.); Tel.: +30-2651-008542

Received: 29 March 2019; Accepted: 30 April 2019; Published: 5 May 2019

Abstract: Data security plays a crucial role in healthcare monitoring systems, since critical patient information is transacted over the Internet, especially through wireless devices, wireless routes such as optical wireless channels, or optical transport networks related to optical fibers. Many hospitals are acquiring their own metro dark fiber networks for collaborating with other institutes as a way to maximize their capacity to meet patient needs, as sharing scarce and expensive assets, such as scanners, allows them to optimize their efficiency. The primary goal of this article is to develop of an attack detection model suitable for healthcare monitoring systems that uses internet protocol (IP) virtual private networks (VPNs) over optical transport networks. To this end, this article presents the vulnerabilities in healthcare monitoring system networks, which employ VPNs over optical transport layer architecture. Furthermore, a multilayer network architecture for closer integration of the IP and optical layers is proposed, and an application for detecting DoS attacks is introduced. The proposed application is a lightweight implementation that could be applied and installed into various remote healthcare control devices with limited processing and memory resources. Finally, an analytical and focused approach correlated to attack detection is proposed, which can also serve as a tutorial oriented towards even nonprofessionals for practical and learning purposes.

Keywords: Denial-of-Service attacks; optical networks; healthcare monitoring; mechanical learning; VPN

1. Introduction

The exponential growth of the Internet and the drastic enhancement of telecommunications [1,2] has rendered the Internet an essential part of everyday life, especially in healthcare activities. Currently, healthcare devices can be easily connected with each other over the Internet in order to process and share useful data through a central processing and controlling server. Remote patient monitoring is becoming more common in the healthcare landscape, with various medical conditions tracked remotely even when patients are not present in the hospital. Wireless medical sensor networks (MSNs) [3–5] are cyber-physical systems (CPSs) [6–9] that have emerged as key building blocks, which provide real-time and ubiquitous remote patient monitoring. Communications are conducted over the Internet, and they are established between medical staff terminals and the devices that monitor the patients [10–12]. Consequently, this type of communication is vulnerable to a variety of cyber-attacks [13–16]. In the past, such attacks have caused enormous problems in healthcare institutes and hospitals. For example, they could destroy patient data, shut down the network, and could even harm the lives of patients under surveillance [17–19]. A technical solution to protect patient data through the Internet is the use of a virtual private network (VPN). A VPN is a concept that virtualizes the private network and utilizes strong security solutions for providing private communications over the public physical network. A VPN is an alternative to a private network or private leased line connection. This idea is very important and should be further enhanced with attack detectors employing mechanical learning. On the other hand, optical VPNs (OVPNs) are regarded as a promising means in modem telecommunications for providing intelligent, flexible, and highly secure infrastructure capable of meeting the most stringent requirements of high-end customers. As such, many hospitals are acquiring their own metro dark fiber connections. This allows them to collaborate with other institutes in a way that maximizes their capacity to meet patient needs, since sharing scarce and expensive assets, such as scanners, allows them to optimize their efficiency.

The integration of optical and wireless access networks, which increases the capacity and mobility of future network architecture, has recently gained much interest. OVPNs provide a secure, high-bandwidth private network that connects end-user sites with a flexible, managed virtual infrastructure over fractional, single, or multiple transparent optical wavelength connections with a wide variety of client interfaces (including Ethernet, optical transport network (OTN), synchronous optical network (SONET), synchronous digital hierarchy (SDH), storage area network (SAN), and video) [20]. Moreover, a VPN is considered a more secure solution than local area network (LAN) networks. Nevertheless, VPNs are not invulnerable, and they can be a target of advanced attacks; therefore, the detection of these kinds of attacks are also studied in this paper.

In this article, we propose our designed VPN over optical transport layer architecture for a remote healthcare monitoring system, while we introduce an attack detection system that uses machine learning. Specifically, a machine learning mechanism (mechanical learning) was employed to recognize and classify network attacks, which was introduced by Wattanapongsakorn et al. in [21]. Our implementation utilizes a packet pre-processing mechanism that performs the aforementioned classification and blocks the Internet Protocol (IP) or the attacked port.

2. System Overview

The proposed scheme aimed at protecting against denial-of-service (DoS) attacks to the remote patient control systems and used an internet protocol security (IPSec)-based VPN service over an optical network. The system under consideration was a multilayer network solution for closer integration of the IP and optical layers. In order to locate the vulnerabilities of a remote patient control system, we developed a model of attack detection by conducting mechanical learning with the use of decision trees. The proposed model could help a user to locate various types of attacks, focusing mainly on flood attacks, and it could be applicable to devices with limited memory and processing resources such as healthcare sensors and devices. It should be emphasized that the proposed scheme was designed as a standalone solution, and it was not a part of an advanced intrusion detection system. Our conducted experiments revealed that such types of attacks were the most frequent in VPNs; they did not require advanced hardware, despite only having good knowledge of the various vulnerabilities of the implemented protocols.

The main components of the proposed model are depicted in Figure 1. This model consisted of a VPN over an optical network. The model was intended for a hospital and provided a dedicated optical channel capacity between two or more locations throughout the optical network. Our VPN over an optical network multilayer model combined IP and optical layers for intelligent design of the optical layer, which resulted in a cross-connected-based architecture. In this way, the hospital would be capable of efficiently switching its traffic by using optical cross-connect (OXC) switches to switch from one wavelength to another as well as from one fiber to another. By using this architecture, hospitals that are connected via optical lines can handle large streams of traffic in core networks compared to IP routers, which operate at the electronic layer and are better suited for more granular switching operations.

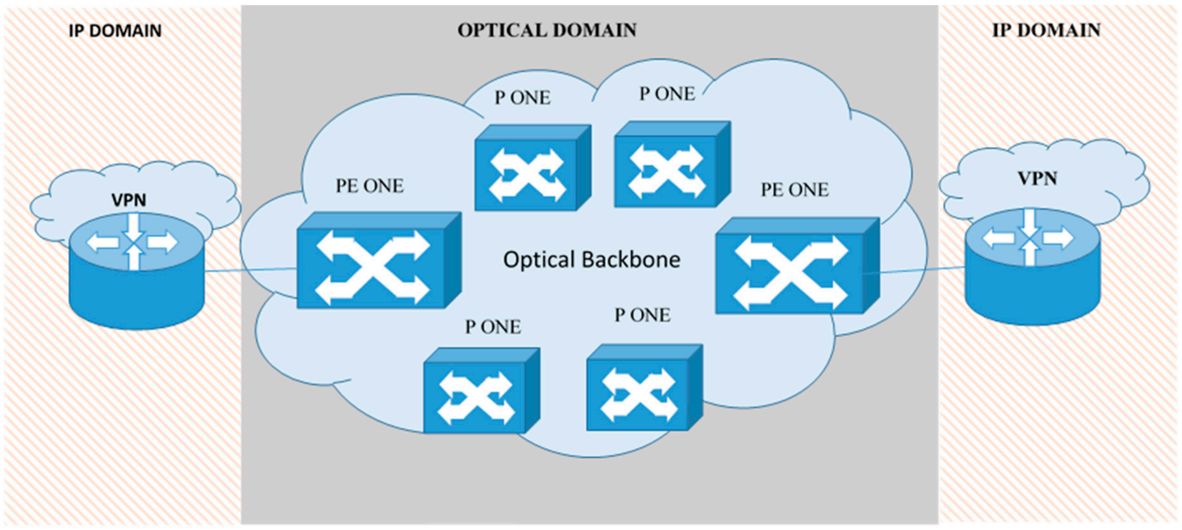


Figure 1. Virtual private network (VPN) over optical network multilayer model.

The optical network backbone can carry voice, video, and traffic data, and healthcare personnel can acquire optical network capacity between locations in small increments (up to the maximum capacity) supported by the hospital's port. In this way, the hospital has great flexibility in upgrading capacity quickly when needed without upgrading the port and local access facility. The bandwidth available in OVPN can be OC-3/STM-1 (155.5 megabits per second), OC-12/STM-4 (622 megabits per second), OC-48/STM-16 (2.5 gigabits per second), OC-192/STM-64 (10 gigabits per second), and gigabit Ethernet.

In this article we present a DoS detector that can prevent intrusions (that may be from outside or inside a VPN), which, if they succeed, could prevent access to all authorized users. These attacks may originate from any point in the network, and they are intended for any point of the network such as the clinic or external unit VPN provider. Moreover, DoS flooding attacks can be applied to our VPN in an IP domain (in a VPN over an optical network multilayer model, which is presented in Figure 1), and it can be rather based on packet header flooding. DoS attacks in an IP domain also expose the optical domain to risk, especially provider edge (PE) optical networking edge (ONE) routers. DoS flooding attacks can disable the IP segment of the PE ONE node. Our detector used an analysis technique for IP networks for the PE ONE node, which was able to catch DoS flooding attacks.

Moreover, we introduced experimental results in order to evaluate the detection times of the mechanical detection algorithm [21] for different rates of flooding packets, which were sent to the attacked entity. The literature has focused only on the acceptance rate of the intrusion detection method. Despite the aforementioned fact, here we try to show that by adapting mechanical learning to intrusion detection methods, we can achieve similar detection performances for both small and large numbers of malicious packets.

All detection methods use a specific threshold of the malicious packets' number that is sent to the attacked entity (see [22,23]). Here, we evaluated the behavior of a mechanical detection algorithm that captured flooding attacks before the network threshold alarm blocked communication. Most detection algorithms do not specify the exact number of malicious packets, but they provide a range. Thus, we hope that our study can help developers understand that the rate of malicious packets is based on the detection algorithm.

In our VPN over an optical network multilayer model, both provider (P) and PE (Figure 1) were network elements such as optimal cross-connect (OXC), pure optical or electronic optical, or SONET/SDH cross-connects. The P ONEs were connected only to optical elements of the hospital provider's network. The IP domain VPN was the same as shown in Figure 2. The IP domain VPN was connected to a PE ONE via several links, whereas each link itself consisted of several channels or subchannels (e.g., wavelength or time slot).

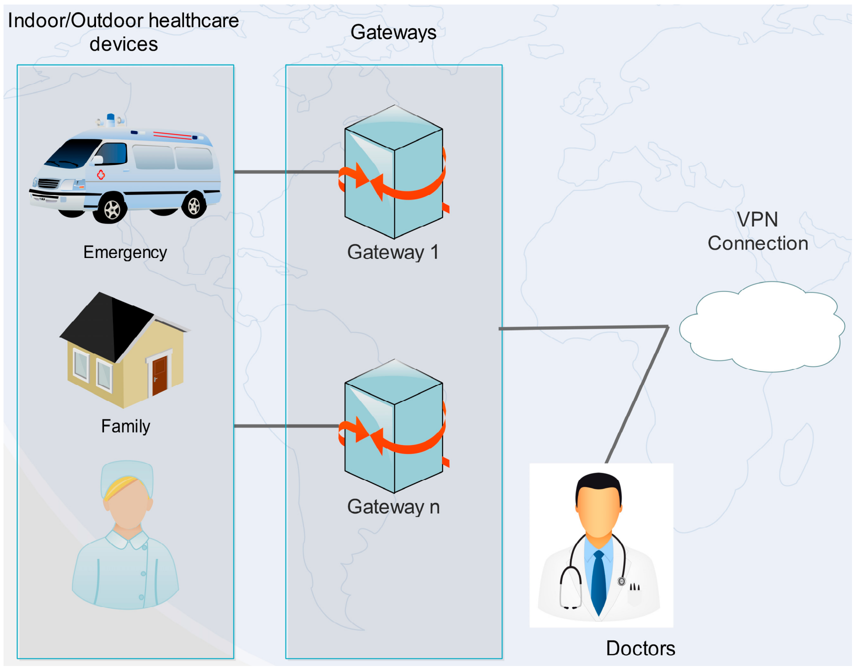


Figure 2. General remote healthcare monitoring system using VPN.

The hospital's back bone optical network must be equipped with the following:

- An IP router network with its point-to-point links served by intelligent optical transport networks.
- Dynamically switched light paths, which can be reconfigured with network changing conditions while they help the optical backbone network establish this virtual point-to-point transport link.
- Signaling protocols (e.g., generalized multi-protocol label switching (GMPLS)) for dynamic provisioning of light paths and automatic restoration.

The user network interface defines an interface between the hospital's VPN routers and the optical network, and it supports the exchange of routing and signaling protocol information between the two layers. The main components of an optical network are the optical or time-division multiplexing connection between two different hospital VPN networks (depicted in Figure 3) in different regions. In Figure 3, a provider is an edge device within a service provider network. A general remote healthcare monitoring system that uses a VPN is shown in Figure 2. Specifically, the healthcare monitoring system is an edge device within a hospital network that provides the interface to the doctor's and indoor/outdoor hospital's personnel domain.

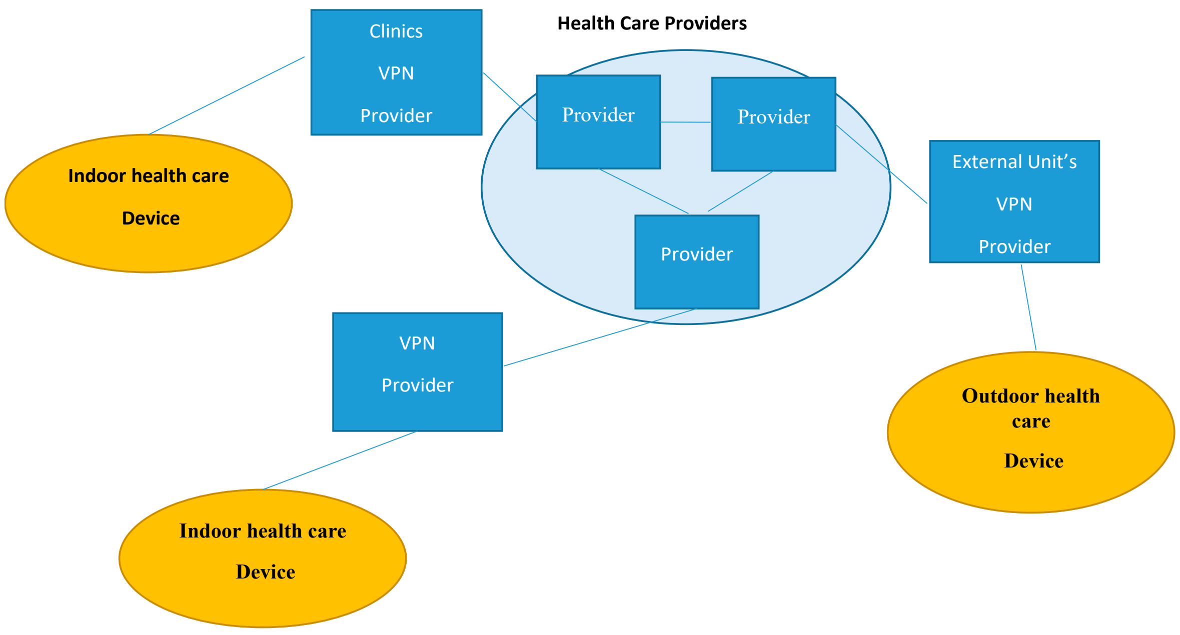


Figure 3. VPN in an IP domain and remote healthcare monitoring system.

Healthcare providers afford the VPN services to hospital personnel. Initially, sensor devices from indoor or outdoor patients send their requests to nearby gateways that forward them to the central processing and controlling server. The latter verifies the requests, and in turn it communicates with gateways and local healthcare personal computers (PCs). The central processing and controlling server also generates and manages VPN client certificates. Patients and doctors communicate with the aforementioned server, which accepts the service requests (Figure 2). The overall security of a generic healthcare monitoring system is further reinforced by establishing a VPN network for communicating between the gateways, doctor computers, and the central processing and controlling server. VPN connections use secure sockets layer (SSL) sessions with bidirectional authentication (i.e., each side must present its own certificate).

The VPN provides remote access service for patients or the doctors when it is impossible to use a leased line or private network. If the logical link fails, then an alternate logical path can be chosen to provide a reliable VPN service. VPN is a combination of tunneling, authentication, integrity, encryption, and access control [24]. However, security is still a major concern [17–19].

3. Threats in Remote Healthcare Control Systems

3.1. Threats

In order to check the vulnerabilities of a remote healthcare control system, the generic architecture shown in Figure 1 was used. Specifically, Figure 1 depicts the generic architecture of a healthcare control system where the network was properly adjusted for facing several implemented attacks against the server that provided VPN services. In this work, we focused on three types of attacks: User datagram protocol (UDP) flood, internet control message protocol (ICMP) flood and transmission control protocol (TCP) synchronous idle mode (SYN) flood.

DoS attacks are one of the most common types of attack on servers. DoS attacks usually reduce the available resources of the victim. In this study, we focused on flood attacks. This kind of attack imposes large computation tasks on the victim machine by flooding it with a huge rate of duplicate packets. In turn, this results in forcing out the victim from the network service for several days.

The Multi-State Information Sharing and Analysis Center (MS-ISAC) in their December 2017 report [25] presented various DoS attacks. In our article we focused on three basic flooding DOS attacks: UDP flood, ICMP flood, and TCP SYN flood as aforementioned. In order to deal with the attacks, we present their main characteristics as they were presented in Reference [25]:

- UDP flood—UDP flood is a type of attack that sends a large number of UDP packets to various random destination ports on a host. This forces the server to process each one, and in most cases respond to each one. This type of attack can quickly lead to the consumption of all available bandwidth.
- ICMP flood—An ICMP flood overthrows the target host with a large number of ICMP packets in an attempt to consume all available bandwidth and deny legitimate access. This attack works well when a large number of sources can send enough ICMP traffic to consume all available bandwidth in the target network.
- SYN flood—A SYN flood attack is one of the most common forms of denial-of-service (DoS) attacks observed by the MS-ISAC report. It is based on a TCP connection sequence where a SYN request opens network communication between a prospective client and the target server. When the server receives a SYN request, it responds by acknowledging the request and holding the communication open, while it waits for the client to acknowledge the open connection. SYN flood attacks occur when an attacker sends a succession of TCP synchronize (SYN) requests to the target host in an attempt to consume enough resources and make the server unavailable for legitimate users. It could exhaust all available server resources and result in denial of service.

3.2. Implementing Attacks

Although our proposed system employed a VPN network, it was still vulnerable to intrusion and DoS attacks. DoS attacks may originate from outside or inside a VPN, preventing access to all authorized users. These attacks may occur at any time-point in the network. During development of our model, it was assumed that the most important part of a VPN was its server, which provides the services, along with its reliability and ability to detect attacks. These attacks could render the server unable to respond to the demands of the connected users. Still, the server is the primary target where an attack could occur, and if the server's connection is cut off from the exterior Internet then all its users' services could become unsustainable. For this reason, we focused on volumetric attacks (also known as flood attacks) aimed at the server and originated from exterior Internet sources. Such kinds of attacks can render the server incompetent in responding to any demand of providing service to VPN users. Analysis of these attacks is of great importance because VPNs are vulnerable to them [26]. Below, the types of applied attacks are reported in detail.

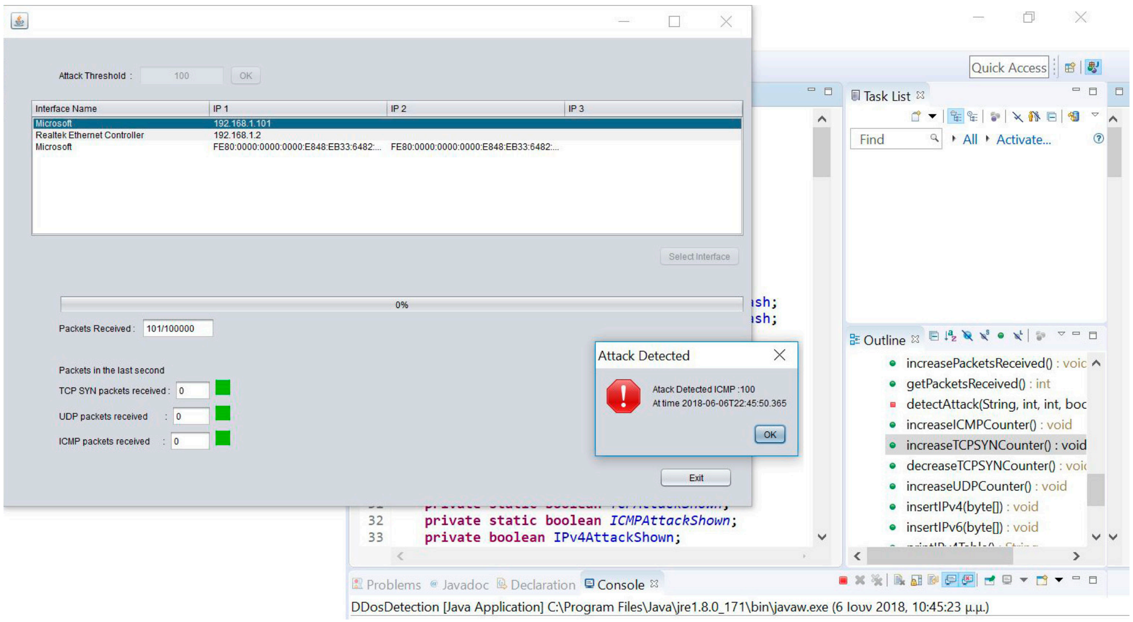
- Regarding ICMP packages, our application checked if the packages were not reply packages. Specifically, the application inspected those packages that were not a response to some information requested by the host server, and then the application's counter was increased.
- The counter of UDP packets increased after every sent packet, while UDP packets did not need an answer or a certain process of initialization and identification.
- In the case of TCP packages, only the reception of multiple TCP SYN packages was investigated. TCP SYN flood is one of the most popular types of attack that uses TCP packets.

It should also be noted that during program execution, the type of attack could be selected, and the information regarding the available IPs for attacking could be acquired based on which available IP was selected; the application selected the source IP that was provided by the network.

4. Attack Detector

The proposed application was a lightweight application (that was developed in Java), which provided a graphical user interface and could run on any hardware device intended for health monitoring with limited resources. Initially, the user had to configure the application to the operated hardware device by selecting the packet number level, was the threshold of the considered attack.

Moreover, in the configuration phase the user had to define which network device would capture the packets. Figure 4 shows the configuration phase where the user selected 500 packets as a threshold and also depicts how the user can select the wireless network card through the provided graphical interface.



Attack Threshold : 100

Interface Name	IP 1	IP 2	IP 3
Microsoft	192.168.1.101		
Realtek Ethernet Controller	192.168.1.2		
Microsoft	FE80:0000:0000:0000:E848:EB33:6482:...	FE80:0000:0000:0000:E848:EB33:6482:...	

0%

Packets Received : 101/100000

Packets in the last second

TCP SYN packets received : 0 🟩

UDP packets received : 0 🟩

ICMP packets received : 0 🟩

Attack Detected

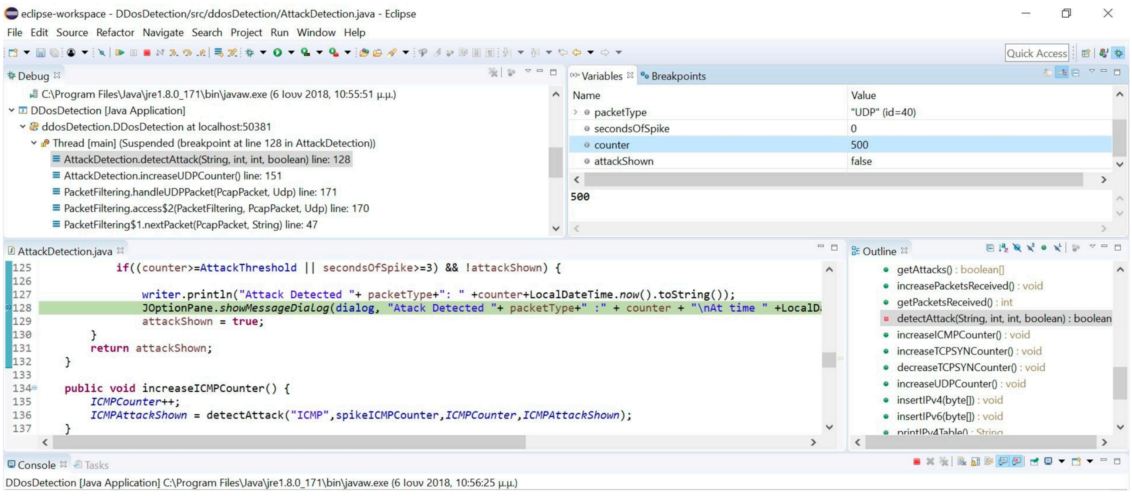
Atack Detected ICMP :100
At time 2018-06-06T22:45:50.365

```
    private static boolean ICMPAttackShown;
    private boolean IPv4AttackShown;
```

DDosDetection [Java Application] C:\Program Files\Java\jre1.8.0_171\bin\javaw.exe (6 Ιουν 2018, 10:45:23 μ.μ.)

OK
Select Interface
Exit
Quick Access
Task List
Find
All
Activate...
Outline
increasePacketsReceived() : void
getPacketsReceived() : int
detectAttack(String, int, int, boolean
increaseICMPCounter() : void
increaseTCPSYNCounter() : void
decreaseTCPSYNCounter() : void
increaseUDPCounter() : void
insertIPv4(byte[]) : void
insertIPv6(byte[]) : void
Problems
Javadoc
Declaration
Console

(**a**) Graphical user interface (GUI) of attack detector.



eclipse-workspace - DDosDetection/src/ddosDetection/AttackDetection.java - Eclipse

```
C:\Program Files\Java\jre1.8.0_171\bin\javaw.exe (6 Ιουν 2018, 10:55:51 μ.μ.)
DDosDetection [Java Application]
  ddosDetection.DDosDetection at localhost:50381
    Thread [main] (Suspended (breakpoint at line 128 in AttackDetection))
      AttackDetection.detectAttack(String, int, int, boolean) line: 128
      AttackDetection.increaseUDPCounter() line: 151
      PacketFiltering.handleUDPPacket(PcapPacket, Udp) line: 171
      PacketFiltering.access$2(PacketFiltering, PcapPacket, Udp) line: 170
      PacketFiltering$1.nextPacket(PcapPacket, String) line: 47
```

Name	Value
packetType	"UDP" (id=40)
secondsOfSpike	0
counter	500
attackShown	false

500

```
            if((counter>=AttackThreshold || secondsOfSpike>=3) && !attackShown) {

                writer.println("Attack Detected "+ packetType+": " +counter+LocalDateTime.now().toString());
                JOptionPane.showMessageDialog(dialog, "Atack Detected "+ packetType+" :" + counter + "\nAt time " +LocalD
                attackShown = true;
            }
        return attackShown;
    }

    public void increaseICMPCounter() {
        ICMPCounter++;
        ICMPAttackShown = detectAttack("ICMP",spikeICMPCounter,ICMPCounter,ICMPAttackShown);
    }
```

DDosDetection [Java Application] C:\Program Files\Java\jre1.8.0_171\bin\javaw.exe (6 Ιουν 2018, 10:56:25 μ.μ.)

File
Edit
Source
Refactor
Navigate
Search
Project
Run
Window
Help
Quick Access
Debug
Variables
Breakpoints
AttackDetection.java
Outline
getAttacks() : boolean[]
increasePacketsReceived() : void
getPacketsReceived() : int
detectAttack(String, int, int, boolean) : boolean
increaseICMPCounter() : void
increaseTCPSYNCounter() : void
decreaseTCPSYNCounter() : void
increaseUDPCounter() : void
insertIPv4(byte[]) : void
insertIPv6(byte[]) : void
Console
Tasks

(**b**) UDP packet with counter over threshold that we consider as attack

Figure 4. GUI of attack detector and UDP packet with the counter over the threshold that we consider as an attack.

The detection method was based on a machine learning detection scheme that was introduced by Wattanapongsakorn et al. in [21], and it combined mechanical learning mechanisms for classifying network attacks. Our implementation consisted of three phases: (i) the packet preprocessing mechanism, (ii) the identification and categorization of the attack, and (iii) the blocking mechanism of the attacked IP/port. Specifically, in the identification phase, our scheme used the already processed data derived from the packet pre-processing mechanism, and it used various machine learning algorithms [27] to mine the processed data and identify the type of attack. The machine learning methods included ripple rule, random forest, decision tree, and Bayesian network (available in [27]). Our experiments revealed that the decision tree machine learning technique worked well, mainly for untrained or unknown network environments. The attack categorization phase could be based on more than one machine learning algorithm for mining processer packet file data, while the result was saved and sent to the blocking mechanism [27–29].

After the user selected the proper network device, packet logging started. In this graphical environment, there was also a bar that shows the percentage of captured packets. Figure 4 shows that 100,000 packets were captured while indicating the number of packets that corresponded to each protocol (up to the previous second). Finally, there was an indicator that showed whether an attack was detected for each protocol or not. These attacks were detected by a decision tree that employed mechanical learning. The program learned its behavior for each type of protocol and detected the attacks accordingly. For the aforementioned implementation, the traffic for each of these packet types was observed. Then, if the observed packet traffic exceeded 500% of the mean observed traffic for more than three seconds, or the momentum of the protocol was greater than the level of the packet numbers set for each protocol type from the user, then an attack was detected and immediately presented to the server user. Additionally, file recording was conducted that included the time and type of attack. In addition to packet type detection, a corresponding mechanism was implemented to identify and record the IP addresses of the traffic's origin. So, if increased traffic was being observed from a particular address, the traffic was recorded in the same log file as the detected attack.

5. Implementation Details

In our experiments, we utilized three main procedures. The first procedure identified the type of attack, the second captured and analyzed the network traffic, and the final procedure applied the attack and visualized the detection result.

5.1. Implementation Tools

The application was developed using Eclipse platform, as it constituted a platform for developing web and other applications. As a tool for developing applications, it did not offer sufficient functionalism. However, the fact that distinguished it among other tools was the capability of using various plugins. It offered a simple graphic environment, and it could be executed in a variety of functional systems along with plugins. Specifically, we used the EclipseOxygen2 platform and EclipseIDE plugin for JavaDevelopers as the attack detector. Also, in order to monitor traffic in the network, usage of Winpcap software was essential for capturing and filtering transmitted packets. Moreover, for developing the application, the open-source library Jnetpcap was used, which provided the interface for capturing and monitoring the packets that passed through a network device. Furthermore, the implemented scenario assumed that a computer/user performed the attack by sending packets using Linux. From the attacker's side, we executed a program called Pentmenu that created the attacks.

5.2. Attack Detection

We implemented a user interface that initialized and adjusted all necessary operational variables of the graphical environment, and we conducted needed changes to the latter environment. We initially stored IP addresses from which we received packages. Consequently, as many packages were received from a specific IP address, we were able to detect an attack. We used a hash function to store the IPs. Our application counted the size and the number of the received packages. Furthermore, our main detection procedure checked whether the packets, that were priorly measured, were either larger than the user's predefined threshold or they exceeded the predefined threshold. In the latter, the user was notified, and a flagged value changed in order to not estimate the attack for one second. In TCP packets, when an acknowledgement (ACK) packet came in response to a SYN/ACK, then the number of packages was reduced. Additionally, a file was initialized and written in every detected attack, which was conducted according to the protocol while the time of the incident was also being recorded. Figure 4 depicts a UDP packet detection that exceeded the initial threshold value. A flow chart of the proposed scheme is depicted in Figure 5. The theory of operation from "program initialization" to "end of program" is presented in the following paragraphs.

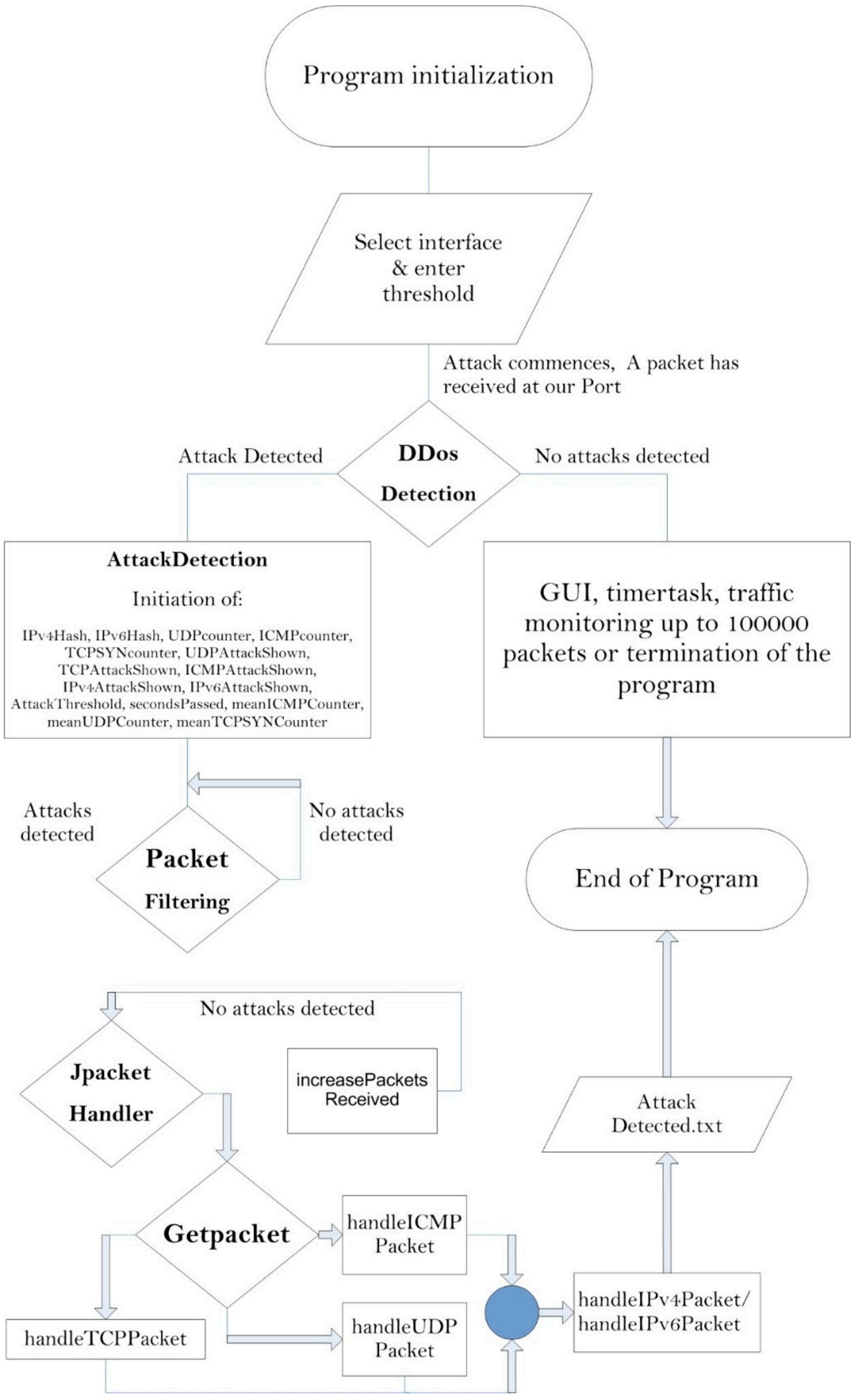

Program initialization
Select interface & enter threshold
Attack commences, A packet has received at our Port
Attack Detected
DDos Detection
No attacks detected
AttackDetection
Initiation of:
IPv4Hash, IPv6Hash, UDPcounter, ICMPcounter, TCPSYNcounter, UDPAttackShown, TCPAttackShown, ICMPAttackShown, IPv4AttackShown, IPv6AttackShown, AttackThreshold, secondsPassed, meanICMPCounter, meanUDPCounter, meanTCPSYNCounter
GUI, timertask, traffic monitoring up to 100000 packets or termination of the program
Attacks detected
Packet Filtering
No attacks detected
End of Program
No attacks detected
increasePackets Received
Jpacket Handler
Attack Detected.txt
Getpacket
handleICMP Packet
handleTCPPacket
handleUDP Packet
handleIPv4Packet/ handleIPv6Packet

Figure 5. Flow chart of the proposed scheme.

5.3. *Analyzing the Network Traffic*

Hereafter, the object-oriented design of the proposed application is described. The PacketFiltering class was the main class of the utilized code. It contained the capture methods of the packages and their analyses. It also contained the instantiation variables of the AttackDetection class as well as the device's information from which the packets were captured. Finally, it contained the JpacketHandler function, which was called each time a packet was captured. This function inspected packet header types and called the corresponding capture function to processes the specific malicious data. Another call was the increasePacketsReceived function, which increased the number of packets received by one. Additionally, the JpacketHandler function called the increasePacketsReceived each time to increase the number of received packets by one. The class constructor initiated only the object of the attackDetection class. The getAttackthreshold function received the potential number of packages from the user, where an attack was considered above this limit.

Use of the functions that returned the ad objects and the device must also not be omitted. Specifically, the SelectItfcToCapture function called the findAllDevs function of the Jnetpcap library, which returned an array list of all possible network devices from which we could receive packets. Then, the return value of the function was checked, and all devices were recorded as well as their IPv4 and IPv6 addresses. Afterwards, the user used this information in order to choose which device wanted to capture and insert the corresponding device variable on this device.

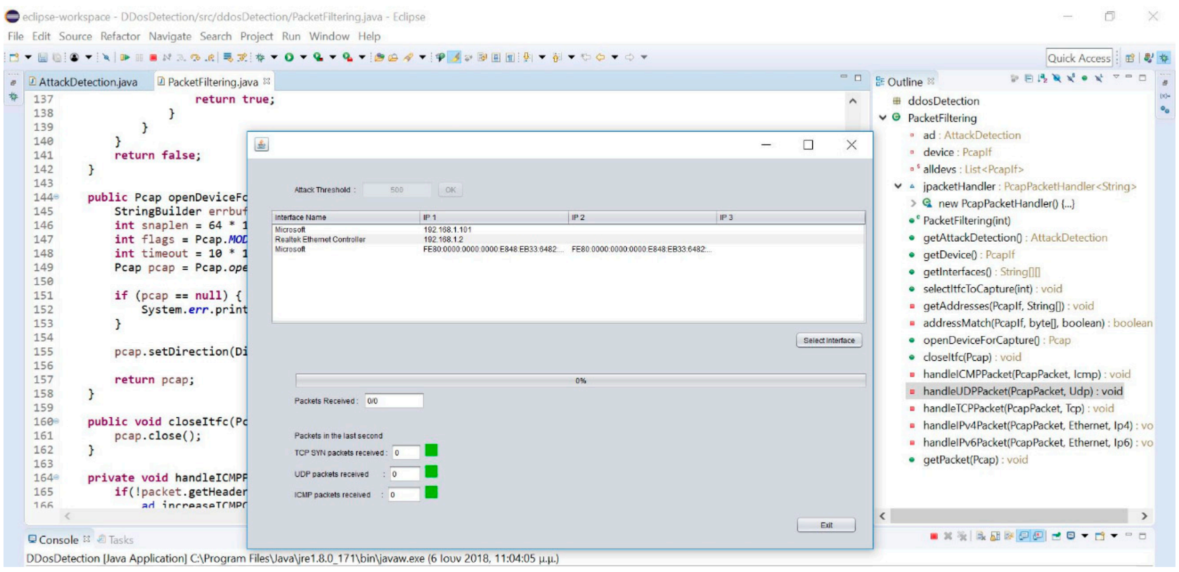
The addressMatch function was used by the functions, which checked whether IPv4 and IPv6 packets were inbound or outbound. This function had Boolean argument that corresponded to the address we wanted to control (e.g., IPv4 or IPv6). The openDeviceForCapture function essentially opened (initiated) the device so that we could read from it. It set the necessary snaplen (snap length) flags timeout variables to pass to the openLive jnetpcap library, which initiated the device so that we could get packets from it. Then, with the setDirection function we adjusted the conditions that would only receive the incoming packets and return a Pcap type variable necessary to obtain the packets. Also, the CloseItfc function closed (terminated) the device we opened so that there were no memory leaks. Additional functions for each header type are presented below:

- *handleICMPPacket*: if packages were not a response to some information requested by the host server, then the function increased the counter (see Section 3.1—ICMP flood).
- *handleUDPPacket*: The counter was incremented by one each time (UDP flood).
- *handleTCPPacket*: The counter was incremented by one for each SYN/ACK packet that we received. On the contrary, the counter decreased by one for every acknowledged received ACK packet (see Section 3.1—TCP SYN flood).
- *handleIPv4Packet/handleIPv6Packet*: This checked whether the source address of the packet was the same as the device's address that we had, and if it was not, then this function increased the counter by one.

Finally, the getPacket function called the loop library function to receive and processes one packet according to the jnetpcaphandler function. Here, the jnetpcaphandler function could process more than one package, but in the current implementation it could lead to thread blocking.

5.4. Main Detection Result

The DDoSDetection class was the main class that ran the entire program. It implemented two objects, one being the packetFiltering class while the other was the graphical user interface (GUI). Figure 6 shows the graphical interface and the network device selection. This class also implemented a counter that worked for one second each time. It defined the function–variable timertask, which determined the counter's action after every count. After the second pass of the counter, the colors were updated in the graphical environment to show a discovered attack. The graphical interface used the counters' values and refreshed the counters' graphical representation. In the beginning, the Main funtion initialized the two objects by calling New and then waited until the user entered the bound value in the graphical environment. The application filled the device board and waited for the user to select the source device of the received packets. Afterwards, the application opened the device and entered an endless loop as it started the counter, which ran until it counted 100,000 packets. The application could be also terminated by the user through the graphical environment. When the loop ended, it closed the record, stopped the counting, and closed (terminated) the device.


eclipse-workspace - DDosDetection/src/ddosDetection/PacketFiltering.java - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
AttackDetection.java
PacketFiltering.java
Outline
Console
Tasks

```
                return true;
            }
        }
    }
    return false;
}

public Pcap openDeviceFc
    StringBuilder errbuf
    int snaplen = 64 * 1
    int flags = Pcap.MOD
    int timeout = 10 * 1
    Pcap pcap = Pcap.ope

    if (pcap == null) {
        System.err.print
    }

    pcap.setDirection(Di

    return pcap;
}

public void closeItfc(Pc
    pcap.close();
}

private void handleICMPF
    if(!packet.getHeader
        ad.increaseICMPC
```

Attack Threshold : 500 OK

Interface Name	IP 1	IP 2	IP 3
Microsoft	192.168.1.101		
Realtek Ethernet Controller	192.168.1.2		
Microsoft	FE80:0000:0000:0000:E848:EB33:6482:...	FE80:0000:0000:0000:E848:EB33:6482:...	

Select Interface

0%

Packets Received : 0/0

Packets in the last second

TCP SYN packets received : 0 🟩

UDP packets received : 0 🟩

ICMP packets received : 0 🟩

Exit

```
ddosDetection
PacketFiltering
    ad : AttackDetection
    device : PcapIf
    alldevs : List<PcapIf>
    jpacketHandler : PcapPacketHandler<String>
        new PcapPacketHandler() {...}
    PacketFiltering(int)
    getAttackDetection() : AttackDetection
    getDevice() : PcapIf
    getInterfaces() : String[][]
    selectItfcToCapture(int) : void
    getAddresses(PcapIf, String[]) : void
    addressMatch(PcapIf, byte[], boolean) : boolean
    openDeviceForCapture() : Pcap
    closeItfc(Pcap) : void
    handleICMPPacket(PcapPacket, Icmp) : void
    handleUDPPacket(PcapPacket, Udp) : void
    handleTCPPacket(PcapPacket, Tcp) : void
    handleIPv4Packet(PcapPacket, Ethernet, Ip4) : vo
    handleIPv6Packet(PcapPacket, Ethernet, Ip6) : vo
    getPacket(Pcap) : void
```

```
DDosDetection [Java Application] C:\Program Files\Java\jre1.8.0_171\bin\javaw.exe (6 Ιουν 2018, 11:04:05 μ.μ.)
```

Figure 6. GUI and network devices.

As several DDoS attack tools and traffic generators have been proposed, they are very difficult to categorize. Nevertheless, the authors in [26] provided a good survey and taxonomy. The majority of DDoS attack tools are part of intrusion detection systems and firewalls that are designed for specific operating systems; they cannot be applied to hardware with limited resources. Here we designed an application for detecting only some popular flooding DoS attacks. It was, at the same time, a standalone solution and could be applicable in any device with limited computational and memory resources.

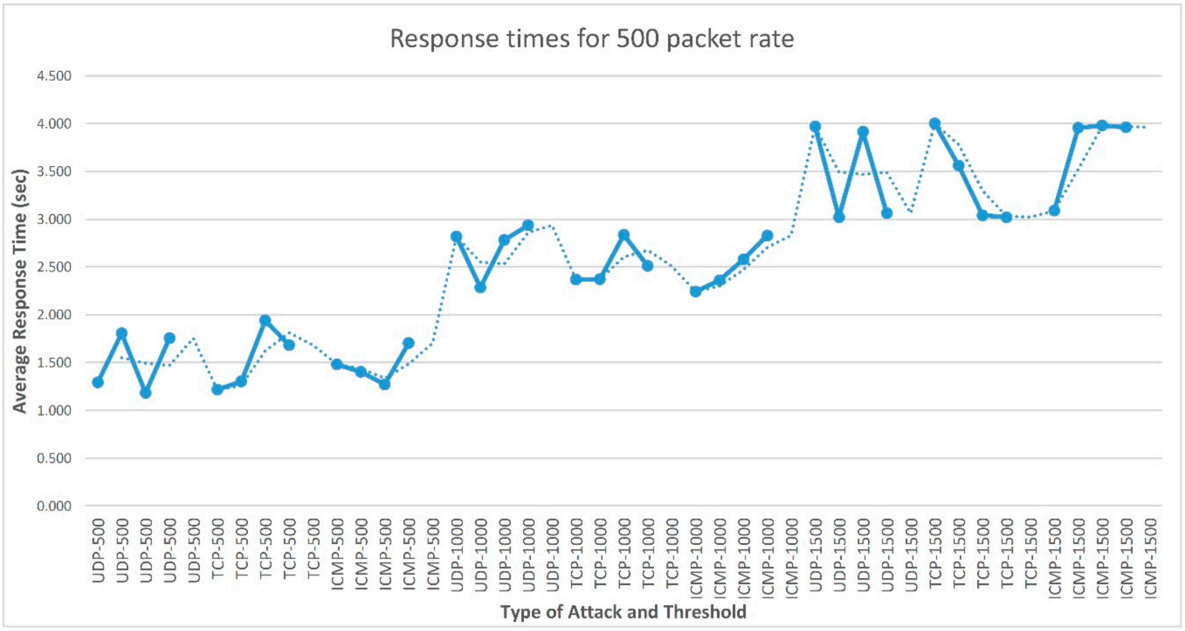
6. Experimental Results

In the beginning, we delimited the number of packets that were considered attacks from a number and above. Then, our real time detector (every second detection) checked the traffic in order to trace packets that exceeded the predetermined limit. As soon as these attacks were detected, our system notified us suitably with essential information including date, hour, type of attack, and the location of the attack such as the closure or the end of traffic control (100,000 packets). Moreover, the proposed system was adjusted to create a log file for further study and confrontation of an intruder.

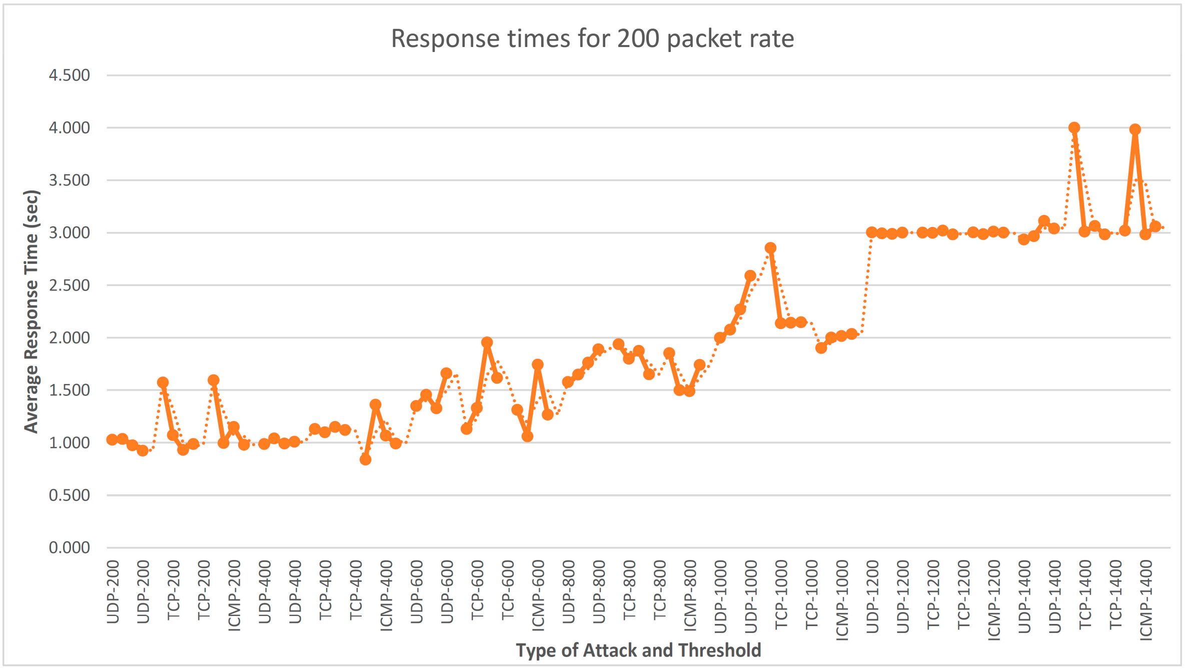
Finally, we used the application in real-time conditions in order to test its behavior relevant to capturing various attacks. Different packet rates were used. Specifically, the application started with 200 malicious packets, and then this size was increased by 200 packets until 1400 packets were reached (see Table 1). The second experiment used 500 malicious packets, and they were increased by 500 in order to reach 1500. We took a sample of the log file that our detector maintained. We started from a small size of packets (200 or 500) and increased to a limit of 1400–1500 packets. While performing the attacks, we observed that for a small number of packets, the detector almost immediately (close to one second) detected the attacks, and when we increased the packets to the predetermined limit, the detection lasted longer (close to two to three seconds). It was apparent that our detector had a quicker response when the size of the malicious packets was smaller. Figure 7 shows that the response times were better when we increased the number of packets by 200 compared to an increase of 500.

Table 1. Response times of the increasing rates of 200 and 500 packets and the average response times for 200 and 500 packet rates.

Type of Attack-Threshold	**Response Time (s)**			
	Response 1	**Response 2**	**Response 3**	**Response 4**
Response Times with 500 Packets of Increasing Rate				
UDP-500	1.291	1.806	1.182	1.755
TCP-500	1.215	1.302	1.941	1.682
ICMP-500	1.480	1.401	1.269	1.702
UDP-1000	2.817	2.283	2.782	2.936
TCP-1000	2.367	2.371	2.836	2.511
ICMP-1000	2.241	2.360	2.577	2.829
UDP-1500	3.969	3.020	3.914	3.062
TCP-1500	4.000	3.560	3.039	3.020
ICMP-1500	3.088	3.955	3.980	3.961
Response Times with 200 Packets of Increasing Rate				
UDP-200	1.028	1.035	0.975	0.923
TCP-200	1.574	1.072	0.932	0.988
ICMP-200	1.596	0.997	1.150	0.979
UDP-400	0.987	1.040	0.992	1.010
TCP-400	1.130	1.100	1.150	1.120
ICMP-400	0.838	1.362	1.068	0.992
UDP-600	1.350	1.457	1.327	1.661
TCP-600	1.130	1.330	1.954	1.616
ICMP-600	1.314	1.060	1.743	1.266
UDP-800	1.579	1.648	1.764	1.889
TCP-800	1.938	1.799	1.874	1.652
ICMP-800	1.853	1.500	1.491	1.741
UDP-1000	2.000	2.076	2.270	2.589
TCP-1000	2.855	2.138	2.143	2.147
ICMP-1000	2.000	2.001	2.017	2.036
UDP-1200	3.004	3.000	2.989	3.001
TCP-1200	3.000	2.998	3.020	2.985
ICMP-1200	3.004	2.986	3.010	3.000
UDP-1400	3.000	2.967	3.112	3.039
TCP-1400	4.000	3.010	3.064	2.985
ICMP-1400	3.020	4.000	2.985	3.060
Type of attack and threshold	**Average Response Time for 500 packet rate (s)**			
UDP-500	1.0000			
TCP-500	1.2280			
ICMP-500	1.1704			
UDP-1000	2.1636			
TCP-1000	2.0170			
ICMP-1000	2.0014			
UDP-1500	2.7930			
TCP-1500	2.7238			
ICMP-1500	2.9968			
Type of attack and threshold	**Average Response Time for 200 packet rate (s)**			
UDP-200	0.990			
TCP-200	1.142			
ICMP-200	1.181			
UDP-400	1.007			
TCP-400	1.125			
ICMP-400	1.065			
UDP-600	1.449			
TCP-600	1.508			
ICMP-600	1.346			



(a)



(b)

Figure 7. Response times for (**a**) 500 and (**b**) 200 packet rates.

As it can be observed, a larger number of malicious packets could provide better detection performance since it could rarely influence non-suspicious packets. The results in Table 1 showed that mechanical learning data mining methodology had similar detection times to smaller and larger numbers of malicious packets.

Moreover, Table 1 presents the response time for 100 attacks and the average response times for three different scenarios (for different thresholds).

7. Conclusions

In this article a proposed application is presented that is a lightweight implementation of an attack detector. This detector can be installed in various healthcare devices, which use IP VPN over an OTN, and they have a lack of processing and memory capabilities. It is based on machine learning mechanisms in order to recognize and classify network attacks. Another goal of this paper was to present a well-studied approach correlated to attack detection with the additional benefit of serving as a tutorial for practical and learning purposes.

Moreover, significant results that we deduced from development of the proposed application were that with the exacerbation of global network attacks, the demand for their identification increased. In order for the attacks to be recognized and addressed effectively, there should also be proper choice of security mechanisms and their protocols, which need to be regularly updated and monitored (for any changes in the flow of packets between users). For example, attacks on home users can cause non-fatal damage (e.g., deletion of a file or computer slow down), but in the case of a company, it could cost a few thousand to millions of euros depending on the size and the extent of the damage. This is the reason that an ideal solution to a securely built network, such as a VPN and OVPN, should include in-depth knowledge of its architecture. It should also, like our proposed software, be able to easily detect threats without an enormous network load (in terms of slowing packet transmission and requiring enormous signal processing). Consequently, the software could be applied to a healthcare system that, nevertheless, should include proper policies such as access rights, network sharing, and security rules. We also concluded that security protocols played a very important role in communication, as their lack of information or proper implementation increased the risk of attack to a network where our proposed software detector was implemented. With regards to the application, as it is quoted, it identifies three types of attacks. As our application used a significant number of various machine learning algorithms (ripple rule, random forest, decision tree, and Bayesian network) for accomplishing detection results in a very short time, this would probably lead to a new and a more enhanced algorithm after extensive testing (under real conditions) and a consequently updated application in the near future. In the future, it could also identify other types of attacks, and we could even develop a mechanism to block enormous numbers and types of attacks. The software could also include teal-time processing with a vastly applied receiver operating characteristic analysis [30,31]. The latter is vastly implemented in radar technology and in interdisciplinary applications. It could definitely use the probabilistic criterions and tools of the related theory in order to "amplify" the detection mechanisms, and in this way, to evolve our detection software. Furthermore, this kind of developed software could also be implemented in leadership in energy and environmental design (LEED) buildings [32] and could be definitely utilized in conjunction with advanced wireless network technologies.

Please be informed that the source codes are available upon request (schrono@cc.uoi.gr).

Author Contributions: Conceptualization, V.L., V.K. and S.K.C.; Formal analysis, V.L. and V.K.; Funding acquisition, V.L., V.K., S.K.C., D.T., V.C. and K.P.P.; Investigation, V.L. and V.K.; Methodology, V.L., V.K., S.K.C., D.T., V.C. and K.P.P.; Project administration, V.L., V.K. and S.K.C.; Resources, V.L., V.K., S.K.C., D.T., V.C. and K.P.P.; Software, V.L. and V.K.; Supervision, V.L. and S.K.C.; Validation, D.T., V.C. and K.P.P.; Visualization, V.L., S.K.C., V.C. and K.P.P.; Writing— original draft, V.L.; Writing—review & editing, S.K.C., D.T., V.C. and K.P.P.

Funding: This research received no external funding.

Acknowledgments: We would like to thank the reviewers for their constructive comments along with the editorial team of *Computation* for their support throughout the whole process of the editorial handling.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Chronopoulos, S.; Christofilakis, V.; Tatsis, G.; Kostarakis, P. Performance of Turbo Coded OFDM Under the Presence of Various Noise Types. *Wirel. Pers. Commun.* **2015**, *87*, 1319–1336. [CrossRef]

2. Peppas, K.; Skrivanos, A.; Xenos, E.; Zhang, J.; Kouretas, I.; Chronopoulos, S. Effective Capacity of Fluctuating Two-Ray Channels with Arbitrary Fading Parameters. In Proceedings of the 2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), Kalamata, Greece, 25–28 June 2018; pp. 1–5.
3. Milenković, A.; Otto, C.; Jovanov, E. Wireless sensor networks for personal health monitoring: Issues and an implementation. *Comput. Commun.* **2006**, *29*, 2521–2533. [CrossRef]
4. Shnayder, V.; Chen, B.; Lorincz, K.; Jones, T.; Welsh, M. Sensor networks for medical care. In Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems—SenSys'05, San Diego, CA, USA, 2–4 November 2005; p. 314.
5. Vallejos de Schatz, C.; Medeiros, H.; Schneider, F.; Abatti, P. Wireless Medical Sensor Networks: Design Requirements and Enabling Technologies. *Telemed. e-Health* **2012**, *18*, 394–399. [CrossRef] [PubMed]
6. Lee, J.; Bagheri, B.; Kao, H. A Cyber-Physical Systems architecture for Industry 4.0-based manufacturing systems. *Manuf. Lett.* **2015**, *3*, 18–23. [CrossRef]
7. Zhang, Y.; Qiu, M.; Tsai, C.; Hassan, M.; Alamri, A. Health-CPS: Healthcare Cyber-Physical System Assisted by Cloud and Big Data. *IEEE Syst. J.* **2017**, *11*, 88–95. [CrossRef]
8. Wolf, W. Cyber-physical Systems. *Computer* **2009**, *42*, 88–89. [CrossRef]
9. Wu, F.; Kao, Y.; Tseng, Y. From wireless sensor networks towards cyber physical systems. *Pervasive Mob. Comput.* **2011**, *7*, 397–413. [CrossRef]
10. Rathore, D.; Upmanyu, A.; Lulla, D. Wireless patient health monitoring system. In Proceedings of the 2013 International Conference on Signal Processing and Communication (ICSC), Noida, India, 12–14 December 2013.
11. Yoo, S.; Lee, K.; Baek, H.; Ryu, B.; Chung, E.; Kim, K.; Yi, J.; Park, S.; Hwang, H. Development and User Research of a Smart Bedside Station System toward Patient-Centered Healthcare System. *J. Med. Syst.* **2015**, *39*, 86. [CrossRef] [PubMed]
12. Hossain, M.; Muhammad, G. Cloud-assisted Industrial Internet of Things (IIoT)—Enabled framework for health monitoring. *Comput. Netw.* **2016**, *101*, 192–202. [CrossRef]
13. Lounis, A.; Hadjidj, A.; Bouabdallah, A.; Challal, Y. Secure and Scalable Cloud-Based Architecture for e-Health Wireless Sensor Networks. In Proceedings of the 2012 21st International Conference on Computer Communications and Networks (ICCCN), Munich, Germany, 30 July–2 August 2012.
14. Rajkumar, R.; Lee, I.; Sha, L.; Stankovic, J. Cyber-physical systems. In Proceedings of the 47th Design Automation Conference—DAC '10, Anaheim, CA, USA, 13–18 June 2010.
15. Wang, J.; Abid, H.; Lee, S.; Shu, L.; Xia, F. A Secured Health Care Application Architecture for Cyber-Physical Systems. *J. Control Eng. Appl. Inform.* **2011**, *13*, 101–108.
16. Liagkou, V. A trustworthy architecture for managing cultural content. *Math. Comput. Model.* **2013**, *57*, 2625–2634. [CrossRef]
17. Johansson, G.; Staff, S.; Morbin, T.; Barth, B. Cyber-attack Shuts Down US Regional Hospital's Online System. Available online: https://www.scmagazineuk.com/cyber-attack-shuts-down-us-regional-hospitals-online-system/article/737077/ (accessed on 1 May 2019).
18. Spitzer, J. 11 of the Biggest Healthcare Cyberattacks of 2017. Available online: https://www.beckershospitalreview.com/cybersecurity/11-of-the-biggest-healthcare-cyberattacks-of-2017.html (accessed on 1 May 2019).
19. Defending Hospitals Against Life-Threatening Cyberattacks. 2018. Available online: http://www.wbur.org/commonhealth/2018/04/26/hospital-cybersecurity (accessed on 1 May 2019).
20. Zhang, Z.; Zhang, Y.; Chu, X.; Li, B. An Overview of Virtual Private Network (VPN): IP VPN and Optical VPN. *Photonic Netw. Commun.* **2004**, *7*, 213–225. [CrossRef]
21. Wattanapongsakorn, N.; Srakaew, S.; Wonghirunsombat, E.; Sribavonmongkol, C.; Junhom, T.; Jongsubsook, P.; Charnsripinyo, C. A Practical Network-Based Intrusion Detection and Prevention System. In Proceedings of the 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications, Liverpool, UK, 25–27 June 2012. [CrossRef]
22. Configuring Attack Protection. Available online: https://www.cisco.com/assets/sol/sb/isa500_emulator/help/guide/ag1359280.html (accessed on 12 April 2019).
23. Knowledge Base—SonicWall. Available online: https://www.sonicwall.com/support/knowledge-base/170503279224098/ (accessed on 12 April 2019).

24. Partala, J.; Keraneny, N.; Sarestoniemi, M.; Hamalainen, M.; Iinatti, J.; Jamsa, T.; Reponen, J.; Seppanen, T. Security threats against the transmission chain of a medical health monitoring system. In Proceedings of the 2013 IEEE 15th International Conference on e-Health Networking, Applications and Services (Healthcom 2013), Lisbon, Portugal, 9–12 October 2013; pp. 243–248.
25. Guide to DDoS Attacks. Available online: https://www.cisecurity.org/wp-content/uploads/2017/03/Guide-to-DDoS-Attacks-November-2017.pdf (accessed on 25 March 2019).
26. Saraswathi, S. Mitigating Strategy to Shield the VPN Service from DoS Attack. *Int. J. Cryptogr. Inf. Secur.* **2012**, *2*, 53–63. [CrossRef]
27. Weka 3—Data Mining with Open Source Machine Learning Software in Java. Available online: https://www.cs.waikato.ac.nz/ml/weka/ (accessed on 25 March 2019).
28. NCCIC Understanding Denial-of-Service Attacks|US-CERT. Available online: https://www.us-cert.gov/ncas/tips/ST04-015 (accessed on 25 March 2019).
29. Port Scanning Techniques|Nmap Network Scanning. Available online: https://nmap.org/book/man-port-scanning-techniques.html (accessed on 25 March 2019).
30. Tafiadis, D.; Kosma, E.; Chronopoulos, S.; Papadopoulos, A.; Drosos, K.; Siafaka, V.; Toki, E.; Ziavra, N. Voice Handicap Index and Interpretation of the Cutoff Points Using Receiver Operating Characteristic Curve as Screening for Young Adult Female Smokers. *J. Voice* **2018**, *32*, 64–69. [CrossRef] [PubMed]
31. Tafiadis, D.; Chronopoulos, S.; Kosma, E.; Voniati, L.; Raptis, V.; Siafaka, V.; Ziavra, N. Using Receiver Operating Characteristic Curve to Define the Cutoff Points of Voice Handicap Index Applied to Young Adult Male Smokers. *J. Voice* **2018**, *32*, 443–448. [CrossRef] [PubMed]
32. Chronopoulos, S.; Kosma, E.; Tafiadis, D.; Dimopoulos, D.; Raptis, V.; Karvounis, E.; Angelidis, P.; Kostarakis, P. Reduced Ecological Footprints of Modern Facilities Introducing the Implementation of Advanced Wireless Technologies, and Human Resources' Benefits. *Commun. Netw.* **2018**, *10*, 11–29. [CrossRef]