HHS Public Access

Author manuscript

Int Conf Complex Intell Softw Intensive Syst. Author manuscript; available in PMC 2024 June 19.

Published in final edited form as:

Int Conf Complex Intell Softw Intensive Syst. 2023 ; 176: 188–199. doi:10.1007/978-3-031-35734-3_19.

A Lightweight Botnet Exploiting HTTP for Control Flow Denial on Open-Source Medical Systems

Wei Lu

Department of Computer Science, Keene State College, The University System of New Hampshire, Keene, NH USA 03431

Abstract

The recent emergence of open-source medical cyber-physical systems has rapidly transformed the healthcare industry. This can be attributed to advancements in 3D printing technology and the growing popularity of open-source microcomputer systems like Arduino and Raspberry Pi. However, the increased use of these systems in hospitals has also raised cybersecurity concerns. In particular, new technologies, such as IoT devices and other mobile devices, have posed new challenges in exploiting modern botnets and determining their effectiveness with limited resources. In this paper, we propose a lightweight and full-encrypted cross-platform botnet system that provides a proof-of-concept demonstration of how a botnet attack can block control flow from the syringe pump in a testbed of an IoT medical network. The emphasis is placed on minimal deployment time and resource usage, making this lightweight botnet different from most traditional botnets, thus furthering cybersecurity research in intrusion detection for open-source medical systems.

1 Introduction

The IoMT (Internet of Medical Things) refers to using connected devices and technologies in the healthcare field to collect, transmit, and analyze medical data for various purposes, such as patient monitoring, diagnostics, treatment, and health management. Medical devices in a typical IoMT range in size from tiny implantable medical devices to massive objects such as MRI scanners, and open-source medical devices and clinic laboratory instruments using desktop 3D printers and open-source electronic microcomputer systems, including such fluorescence imaging devices [1], micro-dispensers [2] and syringe pumps [3][4].

The IoMT has facilitated the development of innovative healthcare applications but has also given rise to new security and privacy concerns that could impede its progress. For example, in 2011, there were reports of malicious attacks on insulin pumps [5], while in 2018, Halperin et al. revealed the potential for wireless attacks on FDA-approved Implantable Cardiac Defibrillator (ICD) devices [6].

Although there are many good ideas for security mechanisms in the medical device domain, they still need to develop fully. Most existing security solutions focus on prevention,

wlu@usnh.edu .

which employs authentication, encryption, and trust-based security management to protect commercial wearable, implantable, and portal medical devices [7][8]. However, open-source medical devices are being overlooked. In addition, poorly implemented security mechanisms make it easy for potential attackers to gain remote control of smart medical devices using malware or botnets [9][10]. They can then manipulate sensitive data by injecting false health data or cause malfunctions by flooding the IoMT network with many illegitimate requests.

This paper aims to address the security challenges posed by open-source medical devices. To this end, we manufacture an open-source medical syringe pump prototype using simple 3D printed hardware parts, a Raspberry Pi system, an Arduino microcomputer, and an open-source software program. We then propose and develop a lightweight and full-encrypted cross-platform botnet system that provides a proof-of-concept demonstration of how a botnet attack can block the control flow command sent from the Raspberry Pi to the syringe pump. This botnet can completely block the communication between the microcomputer system and the mechanical pump. It can also cause the pump to dispense an unexpectedly large amount of fluid after the attack is terminated, creating a potential risk of overdose when used at the bedside. Such a zero-day botnet attack may disturb the network traffic pattern of the connected medical devices. Thus, it helps further the cybersecurity research in network traffic analysis and intrusion detection/prevention in the domain where network flows collected from this botnet are publicly available for feature engineering and adversarial machine learning.

The remainder of this paper is structured as follows. Section 2 discusses the concept of botnet attacks. Section 3 describes the prototype of the open-source medical syringe pump, including a step-by-step guide to its manufacturing process. In Section 4, we introduce the centralized botnet system that is based on secure HTTP protocol and is fully encrypted. Section 5 presents the installation process of this botnet system; then, the Distributed Denial of Service (DDoS) attacks using this botnet system are conducted against the syringe pump in a controlled testbed IoT network where relevant network traffic using packet capture tools is captured. Finally, in Section 6, we offer concluding remarks and discuss future work.

2 Concept of Botnet Attacks

A bot is a self-operating software program controlled by a remote operator known as the botmaster for performing malicious activities, often without the knowledge or consent of the victim whose computer it has been installed on. The bot allows the remote operator to take control of the victim’s system and instruct it to carry out malicious tasks, including but not limited to mass spamming, distributed denial of service attacks, click fraud, and distributed computing for password cracking or other types of cybercrime.

There are various methods that the hot uses to establish this network structure. Command and control channels must efficiently deliver orders from the botmaster to individual bots while evading detection by security measures [11][12]. The IRC-based channels are very efficient mainly because of their ease of implementation and the capability to form large networks, thanks to their simple network architecture. However, network traffic monitoring can quickly reveal the messages being exchanged between the server and individual clients,

making detecting botnets based on message content analysis easy. As an alternative, botnets use HTTP traffic for command and control (CC) schemes, as it can provide stealth by using a legitimate communication channel and evading traditional firewall-based security. To avoid detection based on deep packet analysis, packets are frequently encrypted. The communication channel between bots and botmasters can be protected and kept from being identified using robust encryption methods.

According to Feily et al. [13], there are five phases in the life cycle of a botnet. The first phase is called an initial infection; it involves an attacker exploiting a known vulnerability in a target system to infect it with malware, providing the attacker with additional capabilities on the victim's machine. A malicious binary, called secondary injection, will be fetched during the second phase by executing additional scripts or programs. Once the binary is installed, the victim's computer becomes a bot. Then in the third phase of the connection, the bot attempts to connect to the C&C server using various methods, officially joining the botnet once the connection is established. The final phase is to maintain the bots for updating their binaries to defend against new attacks. Furthermore, a simplified way of categorizing the life cycle of a botnet has been described into four phases: formation, command, and control (C&C), attack, and post-attack [14]. The attack phase is when a bot executes malicious actions in response to orders received from the botmaster, while the post-attack phase is akin to the maintenance phase.

Existing botnet detection techniques mainly focus on detecting bot activity during the attack, initial infection, and secondary injection phases. These techniques often use traditional intrusion detection methods, which identify botnets by analyzing the behavior of underlying malicious activities and comparing them to known signatures of attacks. In our study, we introduce and implement a lightweight centralized botnet attack exploiting secure HTTP protocol. Unlike traditional botnet attacks, such as Sink [15], Phatbot (which utilized WASTE command) [16], Nugache [17], and Peacomm (Storm worm) [18], the proposed botnet system has a highly secure communication protocol between the bots and the botmaster, making it challenging to detect during the command and control phase.

3 Manufacturing an Open-Source Medical Syringe Pump

This section provides a step-by-step guide on creating an open-source medical syringe pump using simple hardware components, a Raspberry Pi system, an Arduino microcomputer, and open-source software. A 3D printer with Cura software manufactures the pump's physical parts. The Arduino system controls the pump through a CNC shield, while the Raspberry Pi is the control center to send data commands to the Arduino. Additionally, an open-source program monitors the syringe pump's working process when ejecting fluids from the syringe. Tins study uses a regular Creality Ender 3 printer with a fully open-source resume printing function [19]. Manufacturing one syringe pump set costs approximately $410, cheaper than similar commercial products. Details on the raw materials and their prices can be found in Table 1.

Throughout the manufacturing process, the most demanding aspect of 3D printing is guaranteeing that the filament is correctly positioned and that the bed is leveled. This can

require several attempts with trial and error to obtain an accurate print. A heuristic approach to determine if we have achieved a successful print is to observe if the filament adheres to the bed firmly and does not detach easily.

Figure 1 showcases the produced medical pump, which is capable of dispensing fluids from the syringe.

The medical pump acquires data from the Arduino and the CNC (Computer Numerical Control) shield, which allows the motors to rotate clockwise or counterclockwise [32]. In addition, the Raspberry Pi system is linked to a touchscreen pad, serving as the control center for the medical pump [33]. All the directives to the Arduino are executed through this interface. The Arduino consists of two components, the lower part contains the Arduino, which receives data from the Raspberry Pi, and the upper part comprises the CNC shield, which assists the Arduino in managing the medical pump's motors.

To create the 3D-printed parts, we utilized Cura as our program of choice [34]. Numerous public tutorials can aid beginners in setting up this program on their 3D printer before using it. We employed this program to generate the 3D prints by obtaining premade 3D printed files [33]. We then drag these files into the program and configure the settings to print the file accurately, which is relatively easy to accomplish. Typically, we used Dynamic Quality with 80% infill, and the prints were highly successful. Next, Arduino programming is applied to upload data to the physical Arduino device. Finally, we can download and install a third-party library called *AccelStepper* to transmit data from the Arduino to the CNC shield. While transferring data to the Arduino, it is crucial to ensure that the data-sharing cord is compatible with the serial communication ports of the device. Afterward, the open-source python controlling program is utilized to commence running the motors.

4 Database Schema, Command Format, and Monitoring

The proposed botnet system in the paper is completely encrypted and serves as a proof-of-concept for further research in network traffic analysis and intrusion detection and prevention by monitoring its network flow behaviors. It comprises three key components: an HTTPS web server, an SSL bot server, and an execution shell. The web server is built on Apache2 with *mod_wsgi*, and is responsible for calling a Python script that utilizes the *Flask* module to provide web services. Meanwhile, the execution shell, also built using Python, utilizes a PostgreSQL database shared with the web service. Finally, the bot server connects to the web service to provide updates on its status and to the execution shell to respond to commands.

The PostgreSQL database utilized by this botnet system is specifically designed for monitoring bot activity. It includes a concise set of tables for storing relevant data such as command logs, bot notifications, and account information for the web service such as IP addresses, port numbers, and command id.

A command language was created for network communication, and the syntax used for these commands slightly differs from the syntax used when entering them through the

command shell. This is because the command shell applies automatic transformations to the commands.

The syntax of the commands below represents how they are transmitted between the execution node and the bot server. These commands transmit and execute Python scripts and monitor and identify active bots.

The *put* statement is utilized to carry out simple file transfer operations. First, the unprocessed file data is transmitted with a filename to associate with the file. Upon receiving the command, the bot generates a write handle for the file referenced by the given filename and truncates it according to the associated flag. Subsequently, the file data is written into the file, and the stream is terminated. It is presumed that the file data represents a Python script.

The *execute* command is intended to be executed after a put command. When a *put* command is executed, a file is stored in the execution directory of the bot server and identified by a specific name. The name of the file is determined by the execution node when it issues the *put* command. This file can then be retrieved and executed by name using Python's eval statement.

The *ping* statement confirms that the bot server is still operational. Despite its name, this statement does not generate an ICMP echo request. Instead, it establishes a connection over the same socket for all other bot communication. When the bot receives this command, it dispatches a ping notification to the web server.

It is important to keep track of the number of active bots at any given time. In addition, activity monitoring tools can assist the bot manager in identifying network connectivity problems or software issues. Currently, activity monitoring is carried out on-demand, meaning the execution node must issue a *ping* command. Furthermore, we need to ensure that malicious sources cannot commandeer our bots, so we require a method for our bots to authenticate the commands they receive.

Each bot updates the control server whenever it starts up, performs an activity, or shuts down. For example, an activity is storing a file (in response to a *put* command), executing a file (in response to an *exec* command), or responding to a *ping*. The bot will transmit these notifications to the address specified in *bot.conf*. In addition, each notification includes the port number on which the bot server operates. When the control server receives these notifications, it will update the bot_status table to reflect the appropriate notification time, message, and port.

When a bot receives communication through its listening socket, the first step is verifying the command. To verify the command, MD5 is used to hash the command, which is then transmitted through a GET request to the configured *validate_addr*. The MD5 command hash is sent as this request's 'command' parameter. The web service checks the command_log table when the validated request is received. It computes the MD5 of all commands transmitted within the last 10 seconds. If the supplied hash is found among them, the value *True* will be transmitted in response to the bot. If not, *False* will be sent instead.

This mechanism verifies both the origin of the command (i.e., the execution node) and the timeliness of the command.

5 Experimental Evaluation of Botnet Effectiveness

The prerequisites for our experimental evaluation of the effectiveness of the proposed botnet system include the python program for the web server, execution shell, and bot with several additional modules, including *flask, flask-login*, and *psycopg2*. The control database for the backend runs on PostgreSQL. After installing the server process, an account for the "medibot" user must be created by adding the following line ***local all medibot md5*** to the postgres configuration.

The *medibot* database schema can be created by logging in as a Postgres user, which can be done on UNIX platforms. After the *medibot* user has been created, the database schema can be created using the included schema file called "schema.sql". This is done by executing the command ***$ psql schema.sql --username=medibot***

To install the *medibot* package, we invoke the setup script in the *medibot* directory using the following syntax. The execution shell is a basic Python script that wraps the core functions of the *medibot* package, i.e. ***$ python setup.py install***

After installing the package, we edit the *medibot.conf* file to reflect the database configuration and then test the installation by launching the executor script ***$ python executor.py ../medibot.conf***

The web service installation primarily involves installing and configuring Apache2 (i.e., the apache2 package on Ubuntu). In addition to the base Apache installation, the *mod_wsgi* and *mod_ssl* packages must also be installed. OpenSSL is then applied to generate an SSL private key and certificate. Once the SSL private key and certificate have been generated, we copy the contents of the *Medibot-Web* folder to a web folder managed by Apache, such as /var/www/medibot. After this, we modified the Apache configuration file located at */etc/apache2/apache2.conf* on Ubuntu Linux. The values indicated in angle brackets depend on the actions taken in previous steps and the specific system configuration. To create a link to the *medibot.conf* file created during the "execution shell" setup within the web directory, we need to execute the following command from the web directory where the *medibot* services were copied on UNIX: ***$ ln <path to medibot.conf> medibot.conf***

In addition to the base configuration, the bot code has several prerequisites, including the OpenSSL server and the Python bindings to OpenSSL, *pyopenssl*. The bot packages are installed as part of the *medibot* package installation so that we can run the command *python setup.py install* for the package installation. The next step is to update the bot configuration file to point to the correct web server. Moreover, the botnet system can use either a statically defined port which may or may not be available when the bot server starts up, or a dynamically assigned port which is guaranteed to be available. After completing the configuration settings, we can run it using a command ***$ ./bot server***

During the experimental testing of the botnet system, we discovered that it could successfully disable the control host of a medical syringe pump. However, as shown in Figure 2, before the attack, the communication between the syringe pump and the control host was functioning normally, with commands to set fluid parameters being sent and received without issue.

However, after approximately two minutes of the DDoS attack, the central control system's graphical user interface experienced some responsiveness delays. This created a potential safety hazard as a medical operator may accidentally click the “run” button multiple times due to the slow response time, resulting in an overdose of fluid injected into patients. For example, in our simulation, we observed that if the operator clicked the “run” button three times, it would cause three times the amount of fluid to be injected, increasing the dose from 5mm to 15mm. In addition, with a continuous DoS attack, the syringe pump completely froze after about five minutes, leading to potential undersupply issues.

The Open Argus network monitoring system [35] collected 211,364 instances, each with 13 features described in Table 2. Table 3 visually represents the descriptive statistics for these 13 features.

6 Conclusions and Future Work

This paper investigates a denial of control flow attack on an open-source medical syringe pump system. We first create a functioning system prototype using a 3D printer and open-source microcomputer systems like Arduino, Raspberry Pi, and CNC shields to do this. We then explore the vulnerabilities of the open-source software monitoring system that controls the medical syringe pump. Our proof-of-concept approach shows that the pump could dispense too much fluid, not enough fluid, or completely stop injecting fluid into patients in the event of a DoS attack launched by the proposed lightweight centralized botnet system.

The future work for this research primarily involves (1) differentiating such attacks based on the collected network traffic payloads from both malicious and malware-free environments [36]. This can be achieved using feature selection using clustering [37] or co-clustering [38] and advanced machine learning models for transfer learning with a focus on deep transfer learning (DTL) models to enable the detection of such DoS attacks against medical syringe pumps across various IoMT networking environments; and (2) enhance the performance of the botnet system by automating tasks such as database structure creation, SSL key pair generation, and then integrating the web service with the command shell and execution node.

Acknowledgments.

This research is supported by New Hampshire - INBRE through an Institutional Development Award (IDeA), P20GM103506, from the National Institute of General Medical Sciences of the NIH.

References

1. Nuñez I, Matute T, Herrera R, Keymer J, Marzullo T, Rudge T, Federici F Low Cost and Open Source Multi-fluorescence Imaging System for Teaching and Research in Biology and Bioengineering. PLoS One. 2017 Nov 15;12(11):e0187163. doi: 10.1371/journal.pone.0187163. [PubMed: 29140977]
2. Forman CJ, Tomes H, Mbobo B et al. Openspritzer: An Open Hardware Pressure Ejection System for Reliably Delivering Picolitre Volumes. Sci Rep 7, 2188 (2017). 10.1038/s41598-017-02301-2 [PubMed: 28526883]
3. Wijnen B, Hunt EJ, Anzalone GC and Pearce JM Open-source Syringe Pump Library. Plos One 9, e107216, 10.1371/journal.pone.0107216 (2014). [PubMed: 25229451]
4. "Croatt Group DIY Flow Chemistry Setup – UNC-Greensboro". https://chem.uncg.edu/croatt/flow-chemistry/ retrieved on Mar. 27, 2023.
5. Li CX, Raghunathan A, and Jha NK Hijacking an Insulin pump: Security Attacks and Defenses for a Diabetes Therapy System. 2011 IEEE 13th International Conference on e-Health Networking, Applications and Services, 2011, pp. 150–156, DOI: 10.1109/HEALTH.2011.6026732.
6. Halperin D., et al., Pacemakers and Implantable Cardiac Defibrillators: Software Radio Attacks and Zero-Power Defenses. 2008 IEEE Symposium on Security and Privacy, 2008, pp. 129–142, DOI: 10.1109/SP.2008.31.
7. Yanambaka VP, Mohanty SP, Kougianos E and Puthal D PMsec: Physical Unclonable Function-Based Robust and Lightweight Authentication in the Internet of Medical Things, in IEEE Transactions on Consumer Electronics, vol. 65, no. 3, pp. 388–397, Aug. 2019, DOI: 10.1109/TCE.2019.2926192.
8. Su J, Vasconcellos DV, Prasad S, Sgandurra D, Feng Y and Sakurai K Lightweight Classification of IoT Malware Based on Image Recognition. 2018 IEEE 42nd Annual Computer Software and Applications Conference (COMPSAC), 2018, pp. 664–669, DOI: 10.1109/COMPSAC.2018.10315.
9. Garant D, Lu W “Mining Botnet Behaviors on the Large-scale Web Application Community.” In Proceedings of 27th IEEE International Conference on Advanced Information Networking and Applications, Barcelona, Spain, March 25 - 28, 2013.
10. Lu W, Miller M and Xue L “Detecting Command and Control Channel of Botnets in Cloud” in Lecture Notes in Computer Science (LNCS, volume 10618). Springer Nature, pp. 55–62, ISBN 978-3-319-69154-1, Oct. 2017.
11. Lu W. An unsupervised anomaly detection framework for multiple-connection-based network intrusions. Publisher: Ottawa Library and Archives Canada, ISBN: 9780494147795, 2007.
12. Lu W and Ghorbani A "Bots Behaviors vs. Eluman Behaviors on Large-Scale Communication Networks" Proceedings of 11th International Symposium on Recent Advances in Intrusion Detection (RAID 2008), Lippmann R, Kirda E, and Trachtenberg A (Eds.): RAID 2008, LNCS 5230, pp. 415–416, MIT, Boston, USA 2008.
13. Feily M, Shahrestani A and Ramadass S "A Survey of Botnet and Botnet Detection," 2009 Third International Conference on Emerging Security Information, Systems and Technologies, Athens, Greece, 2009, pp. 268–273, doi: 10.1109/SECURWARE.2009.48.
14. Leonard J, Xu S and Sandhu R "A Framework for Understanding Botnets," 2009 International Conference on Availability, Reliability and Security, Fukuoka, Japan, 2009, pp. 917–922, doi: 10.1109/ARES.2009.65.
15. Sinit, https://www.f-secure.com/v-descs/sinit.shtml retrieved on Mar. 27, 2023.
16. Phatbot, https://www.fortiguard.com/encyclopedia/ips/103350720 retrieved on Mar. 27, 2023.
17. Nugache, https://www.usenix.org/system/files/login/articles/526-stover.pdf retrieved on Mar. 27, 2023.
18. Grizzard JB, Sharma V, Nunnery C, Kang BB, and Dagon D Peer-to-peer botnets: Overview and case study. In proceedings of the 1st USENIX Workshop on Hot Topics in Understanding Botnets, Cambridge, MA 2007.
19. "Creality 3D printer". https://www.amazon.com/Comgrow-Creality-Ender-Aluminum-220x220x250mm/dp/B07BR3F9N6 retrieved on Mar. 27, 2023.

20. "Creality CR Touch Auto Bed Leveling Sensor Kit". https://www.amazon.com/dp/B09P4YKRTD/ref=cm_sw_r_apan_i_6ZGJ0ATJ3JRJB55EKEPY?_encoding=UTF8&psc=1 retrieved on Mar. 27, 2023.
21. "HATCHBOX 1.75mm Cool Gray PLA 3D Printer Filament". https://www.amazon.com/dp/B01511CYFE/ref=cm_sw_r_apan_i_N75E3SG3T7T1CPTMQ26C?_encoding=UTF8&psc=1 retrieved on Mar. 27, 2023.
22. "uxcell® M5x14mm 316 Stainless Steel Metric Fully Thread Hex Socket Cap Screws". https://www.amazon.com/gp/product/B01LJROXK0/ref=ox_sc_saved_title_2?smid=A1THAZDOWP300U&psc=1 retrieved on Mar. 27, 2023.
23. "SanDisk SDSDQM-016G-B35A 16 GB Class 4 MicroSDHC Memory Card with SD Adapter". https://www.amazon.com/gp/product/B004G605OA/ref=ox_sc_saved_title_7?smid=ABYURLNKK9M7V&psc=1 retrieved on Mar. 27, 2023.
24. "Raspberry Pi Touch Screen Display". https://www.amazon.com/gp/product/B0153R2A9I/ref=ox_sc_saved_title_6?smid=A6EGA15UEFYEQ&psc=1 retrieved on Mar. 27, 2023.
25. "5mm to 5mm Aluminum Flexible Shaft Coupling". https://www.amazon.com/gp/product/B01EFFBM4I/ref=ox_sc_saved_title_1?smid=A26373IMBF4DLW&psc=1 retrieved on Mar. 27, 2023.
26. "3D Printer Kits CNC Shield V3.0, Keyestudio R3 Board, Nema 17 Stepper Motor, 4PCS A4988 Driver & USB Cable, Heat Sink, Stepper Motor Controller Shield Kit". https://www.amazon.com/Tangxi-Printer-Stepper-Heatsink-Arduino/dp/B07SBDD4HL retrieved on Mar. 27, 2023.
27. "Linear Ball Bearings, Linear Motion Ball Bearing Bushing for 3D Printer CNC Parts". https://www.amazon.com/Linear-Motion-Bearing-Bushing-Printer/dp/B07K71FWMG/ref=dp_prsubs_1?pd_rd_i=B07K71FWMG&psc=1 retrieved on Mar. 27, 2023.
28. "uxcell 2pcs 6mm x 200mm Metal Machine Turning Tool Rod Bar Lathe Round Stick". https://www.amazon.com/uxcell-Metal-Machine-Turning-Tools/dp/B0BJ7D7V23 retrieved on Mar. 27, 2023.
29. "uxcell a16071500ux0127 M5 x 170 mm 304 Stainless Steel Fully Threaded Rod Bar Studs Fasteners". https://www.amazon.com/Uxcell-a16071500ux0127-Stainless-Threaded-Fasteners/dp/B01M4L8JDC retrieved on Mar. 27, 2023.
30. "Raspberry Pi 3 Model B+ Board". https://www.amazon.com/ELEMENT-Element14-Raspberry-Pi-Motherboard/dp/B07P4LSDYV retrieved on Mar. 27, 2023.
31. "uxcell M3x10mm Thread 304 Stainless Steel Hex Socket Head Cap Screw Bolt". https://www.amazon.com/gp/product/B01MFA9YEP/ref=ox_sc_saved_title_1?smid=A1THAZDOWP300U&psc=1 retrieved on Mar. 27, 2023.
32. Booeshaghi AS, Beltrame E.d.V., Bannon D, et al. Principles of Open Source Bioinstrumentation Applied to the Poseidon Syringe Pump System. Sci Rep 9, 12385 (2019). 10.1038/s41598-019-48815-9 [PubMed: 31455877]
33. "poseidon: Open source bioinstrumentation". https://github.com/pachterlab/poseidon retrieved on Mar. 27, 2023.
34. "Ultimaker Cura". https://ultimaker.com/software/ultimaker-cura retrieved on Mar. 27, 2023.
35. "Argus ra 3.0.8". https://qosient.com/argus/man/man1/ra.1.pdf retrieved on Mar. 27, 2023.
36. Ghorbani A, Lu W and Tavallaee M Detection Approaches, Network Intrusion Detection and Prevention: Concepts and Techniques. Springer Publisher, ISBN-10: 0387887709, pp. 27–53, Oct. 20, 2009.
37. Lu W and Traore I "A New Evolutionary Algorithm for Determining the Optimal Number of Clusters." In Proceedings of IEEE International Conference on Computational Intelligence for Modeling, Control and Automation (CIMCA 2005), Volume 1, pp. 648–653, 2005.
38. Lu W and Xue L "A Heuristic-Based Co-clustering Algorithm for the Internet Traffic Classification," 2014 28th International Conference on Advanced Information Networking and Applications Workshops, 2014, pp. 49–54, doi: 10.1109/WAINA.2014.16.

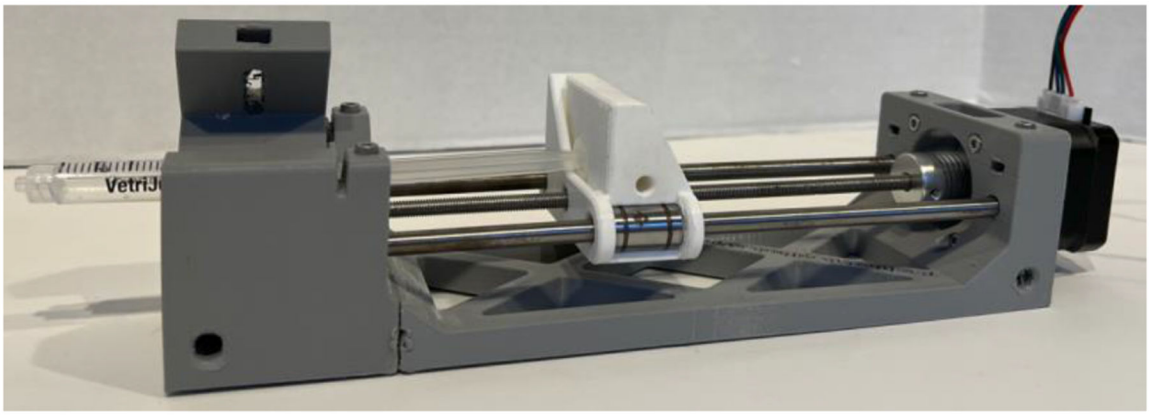

Fig. 1.
The syringe pump

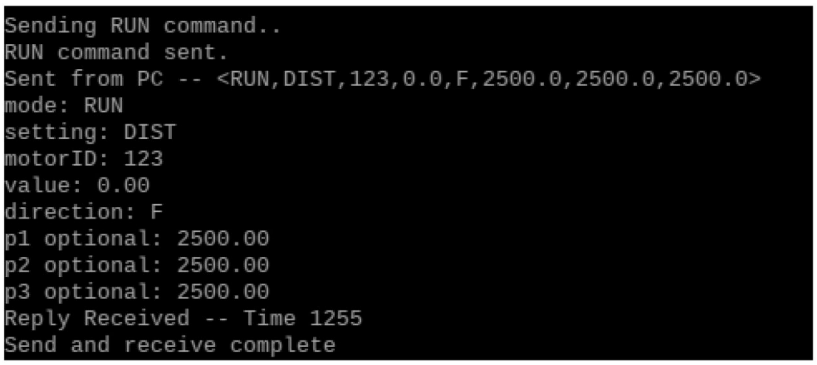

Fig. 2.
Command sent/received between syringe pump and central control system.

Table 1.

Raw materials for manufacturing the open-source medical syringe pump.

Materials	Price
Creality CR Touch Auto Bed Leveling Sensor [20]	$39.0
HATCHBOX 1.75mm Cool Gray PLA 3D Printer Filament [21]	$24.99
uxcell® M5x14mm 316 Stainless Steel Metric Fully Thread Hex Socket Cap [22]	$9.49
SanDisk SDSDQM-016G-B35A 16 GB Memory Card [23]	$6.75
Raspberry Pi 7" Touch Screen Display [24]	$69.99
5mm to 5mm Aluminum Flexible Shaft Coupling [25]	$14.09
CNC Shield V3.0 & Keyestudio R3 Board & Nema 17 Stepper Motor [26]	$36.17
Linear Ball Bearings, Linear Motion Ball Bearing Bushing [27]	$10.62
uxcell 2pcs 6mm x 200mm Metal Machine Turning Tool Rod Bar [28]	$7.49
uxcell a16071500ux0127 M5 x 170 mm 304 Stainless Steel Fully Threaded Rod [29]	$11.51
Raspberry Pi 3 Model B+ Board [30]	$169.99
uxcell M3x10mm Thread 304 Stainless Steel Hex Socket Head Cap Screw Bolt [31]	$9.99

Table 2.

Feature description.

Feature	Description
SrcBytes/DstBytes	Number of bytes from source to destination (or from destination to source)
SrcLoad/DstLoad	Source to destination bits per second (Destination to source bits per second)
SrcPkts	Number of packets from source to destination
DstPkts	Number of packets from destination to source
SrcRate	Number of packets per second from source to destination
DstRate	Number of packets per second from destination to source
Dur	Transaction record total duration
TotPkts	Total transaction packets count
TotBytes	Total transaction bytes
Load	Total transaction bits per second
Rate	Number of packets per second

Table 3.

Descriptive statistics of features for a total of 211,364 data instances.

Feature	Mean	Std.	Min	25%	50%	75%	Max
SrcBytes	286.255	2822.596	0.0	71.0	120.0	180.0	435324
SrcPkts	2.439	22.385	0.0	1.0	1.0	3.0	4890
DstPkts	0.938	10.365	0.0	0.0	0.0	1.0	2411
DstBytes	1052.62	57168.03	0.0	0.0	0.0	142.0	12858140
Dur	0.442	0.801	0.0	0.0	0.0	0.598	4.999
TotPkts	3.377	30.879	1.0	1.0	2.0	3.0	5979
SrcLoad	4765462	55782630	0.0	0.0	0.0	683.77	15168000000
DstLoad	12886.2	292332	0.0	0.0	0.0	0.0	74981780
SrcRate	1551.3	15807.7	0.0	0.0	0.0	1.4	4000000
DstRate	1.983	17.58	0.0	0.0	0.0	0.0	1851.57
TotBytes	1338.9	58882.5	54	74	180	243	13230630
Load	4778348	55782350	0.0	0.0	0.0	684.4	15168000000
Rate	2535.2	17395.6	0.0	0.0	1.25	250	4000000