Performance Evaluation of SNMPv1/2c/3 using Different Security Models on Raspberry Pi

Eric Gamess[1]
MCIS Department
Jacksonville State University
Jacksonville, Alabama, USA

Sergio Hernandez[2]
Information Security
Citibank, New York
New York, USA

***Abstract*—The Simple Network Management Protocol (SNMP) is one of the dominant protocols for network monitoring and configuration. The first two versions of SNMP (v1 and v2c) use the Community-based Security Model (CSM), where the community is transferred in clear text, resulting in a low level of security. With the release of SNMPv3, the User-based Security Model (USM) and Transport Security Model (TSM) were proposed, with strong authentication and privacy at different levels. The Raspberry Pi family of Single-Board Computers (SBCs) is widely used for many applications. To help their integration into network management systems, it is essential to study the impact of the different versions and security models of SNMP on these SBCs. In this work, we carried out a performance analysis of SNMP agents running in three different Raspberry Pis (Pi Zero W, Pi 3 Model B, and Pi 3 Model B+). Our comparisons are based on the response time, defined as the time required to complete a request/response exchange between a manager and an agent. Since we did not find an adequate tool for our assessments, we developed our own benchmarking tool. We did numerous experiments, varying different parameters such as the type of requests, the number of objects involved per request, the security levels of SNMPv3/USM, the authentication and privacy protocols of SNMPv3/USM, the transport protocols, and the versions and security models of SNMP. Our experiments were executed with Net-SNMP, an open-source and comprehensive distribution of SNMP. Our tests indicate that SNMPv1 and SNMPv2c have similar performance. SNMPv3 has a longer response time, due to the overhead caused by the security services (authentication and privacy). The Pi 3 Model B and Pi 3 Model B+ have comparable performance, and significantly outperform the Pi Zero W.**

***Keywords*—Simple network management protocol; SNMP; performance evaluation; benchmarks; raspberry pi**

I. INTRODUCTION

The Simple Network Management Protocol (SNMP) is widely utilized for network monitoring and management. SNMPv1 and SNMPv2c use the Community-based Security Model (CSM), where the community (that can be seen as a password) is exchanged in cleartext between SNMP entities. This basic model of security opens many simple attacks against the protocol. Hence, a new version of SNMP was released and uses the User-based Security Model (SNMPv3/USM). The USM model brings strong authentication and privacy to SNMP. It was designed to work independently of other existing security infrastructures, and utilizes a separate user and key management infrastructure. Unfortunately, the operational cost for deploying another user and key management infrastructure is significant, and network operators have been reluctant in its adoption [1]. To address this issue, the Transport Security Model (TSM) was later added to SNMPv3, and relies on well-accepted secure transport layers such as Secure Shell [2] (SSH), Transport Layer Security [3] (TLS), and Datagram Transport Layer Security [4] (DTLS).

The Raspberry Pi Foundation, a non-profit organization, has released a series of Single Board Computers (SBCs) that have been well-accepted by the community [5][6]. Due to its low cost (for approximately US$10), the Raspberry Pi Zero W (RPi Zero W) is one of the best-selling SBCs of the foundation, and has a 32-bit single-core processor and a WiFi adapter. When more CPU power is required, users might consider the Raspberry Pi 3 Model B (RPi 3B) or the Raspberry Pi 3 Model B+ (RPi 3B+), both with a 64-bit quad-core processor, Ethernet, and WiFi, for approximately US$35.

To facilitate the integration of Raspberry Pi SBCs into network management systems, we carried out an analytical performance analysis of different SNMP versions and security models for three different boards of the Raspberry Pi Foundation: (1) RPi Zero W, (2) RPi 3B, and RPi 3B+. To do so, we installed the agent of Net-SNMP [7], a well-known and comprehensive implementation of the SNMP protocol, on the three SBCs and ran some tests using a benchmarking tool that we developed. For better flexibility, the tool has numerous parameters and reports the "Response Time" defined as the required time to complete an SNMP request/response exchange between a manager and an agent. We performed intensive tests where we varied parameters such as the type of requests, the number of objects involved per request, the security levels of SNMPv3/USM, the authentication and privacy protocols of SNMPv3/USM, the transport protocols, and the versions and security models of SNMP. We think this study might be helpful for network administrators when integrating Raspberry Pis into SNMP-based network management systems.

The rest of the paper is structured as follows. Section II discusses the related work. An introduction to the SNMP protocol and its different versions and security models is made in Section III. We present the benchmark developed and used for the experiments in Section IV. The description of the test environment is done in Section V. Section VI reports and discusses the results of our evaluation of the SNMP protocol in many different scenarios. Finally, Section VII concludes the paper and gives directions for future work.

II. Related Work

Some work has been done to evaluate the performance of SNMP. Andrey, Festor, Lahmadi, Pras, and Schönwälder [8] studied papers related to the evaluation of SNMP, in major research databases such as the IEEE Xplore and the ACM Digital Library. Their goal was to retrieve and classify techniques, approaches, and metrics employed by these studies, to propose a common framework for SNMP performance analysis. Hidalgo and Gamess [9] developed one of the first SNMP agents for Android smartphones with support for SNMPv1 and SNMPv2c. To validate the possibility of integrating them into network management systems, the authors did some performance evaluations of the maximum SNMP traffic that Android smartphones can support in a determined period of time. In their work, Corrente and Tura [10] analyzed the impact of security on SNMP, by considering SNMPv1, SNMPv2c, and SNMPv3/USM. They did experiments in a testbed and reported metrics such as the processing time, number of transactions per minute, CPU usage, and protocol overhead. To more efficiently use SNMP in mobile environments, the study in [11] proposed to add a superimposition model to its architecture. With simulations, the authors supported how the proposed superimposition architecture can improve the performance of SNMPv3/USM. Several studies are focused on comparing the performance of different network management solutions [12-16]. For example, the authors of [12] assessed the performance of SNMP-based and web services-based network monitoring systems. Their analysis was centered around SNMPv1 and SNMPv2c, and they reported results such as bandwidth usage, memory consumption, and roundtrip delay. Another work in this direction was done in [13], where Santos, Esteves, and Granville evaluated the performance of SNMP, NETCONF [17], and RESTful web services for router virtualization management. At the level of SNMP, the authors assessed SNMPv2c and SNMPv3/USM.

The previously mentioned efforts did not consider the new TSM model of SNMPv3. In the specialized literature, just a few projects have included this emerging standard. One of the first evaluations was done by Du, Shayman, and Rozenblit [18], before the publication of the RFCs that introduced the TSM model [19–21]. The authors modified the source code of Net-SNMP [7] and integrated the support of TLS [3] over TCP, for both SNMPv1 and SNMPv3. To demonstrate the viability of such a new development at the level of performance, the research team did some experiments in a testbed environment, and analyzed the performance of SNMPv1, SNMPv3/USM, and their non-standard SNMPv1 and SNMPv3 over TCP with TLS. A few years later, the work in [22] used a similar approach for SNMP over SSH. The authors did a non-standard modification of Net-SNMP [7] to carry SNMPv2c over SSH [2]. In a controlled environment, they assessed the performance of SNMPv2c and SNMPv3/USM, against their non-standard modified version of SNMPv2c over SSH. More recently, Schönwälder and Marinov [1] evaluated SNMPv3/USM and SNMPv3/TSM (with SSH, TLS, and DTLS) in a test environment. The testbed was made of computers connected through Ethernet. They reported metrics such as the response time to execute `snmpget` and `snmpwalk` (retrieving the `ifTable` table [23]) commands, and the bandwidth utilization for `snmpwalk` (retrieving the `ifTable` table [23]). It is worth clarifying that `snmpget` and `snmpwalk` are basic applications shipped with Net-SNMP [7].

According to our search, the unique assessment work that covers SNMPv3/TSM and standard implementations of the protocols is described in [1]. Our paper not only includes SNMPv3/TSM, but we also believe that it will be of interest in the growing community of the Raspberry Pi [5][6].

III. Introduction to SNMP and its Different Versions and Security Models

The Simple Network Management Protocol (SNMP) was initially defined in August 1988 by RFC 1067 [24] as a protocol to monitor and control network devices, and it has been used extensively for over three decades now. SNMP allows configuring network devices remotely, collecting management data, and supporting the dissemination of event notifications [1]. Approved in 1990, SNMP became one of the main network protocols widely used as a de-facto standard by the industry to carry out the monitoring of assets for IP-based networks [25]. Nevertheless, the first version of SNMP, known as SNMPv1, is limited to meet all network management requirements that arise as a consequence of the interconnection complexity among systems, and is exposed to several security threats.

The architectural model of SNMP is straightforward and consists of network management stations, agents, and managed devices. Network management stations execute the applications which monitor and control network elements or managed devices. Agents are responsible for performing the network management functions requested by the network management stations, whereas managed devices may be hosts, gateways, terminal servers, switches, routers, among others.

The second version of SNMP, known as SNMPv2c, is an improvement of SNMPv1 without implementing security features. Neither SNMPv1 nor SNMPv2c can provide authentication, confidentiality, and integrity; therefore, they are exposed to multiple security threats, particularly those associated with authentication and privacy [26].

The third version of SNMP, known as SNMPv3, provides security features to the previous versions by introducing the User-based Security Model (USM), which is used to authenticate entities and provides encryption to secure the communication channel [10]. The authentication is performed using Hashed Message Authentication Code (HMAC) based on techniques such as Message Digest 5 (MD5) as well as Secure Hash Algorithm (SHA), while encryption for privacy is performed using Data Encryption Standard (DES) and Advanced Encryption Standard (AES), which are symmetric algorithms [27]. Also, SNMPv3 introduced a substantial complexity to SNMP architecture, since it implements its own user and key management infrastructure.

A. SNMPv1 and SNMPv2c

Both versions, SNMPv1 and SNMPv2c, rely on the Community-based Security Model (CSM) by which the community's name acts as a password and is transmitted over

the network in cleartext with the message. If the community's name is recognized, then the message should be processed. The use of the community's name without any encryption to verify that the message was sent by a trusted source is inherently insecure since it allows unauthorized individuals to capture it by using a packet analyzer or sniffer, and execute privileged actions. Hence, the security of the SNMP messages is dependent on the security of the channels over which the messages are sent.

SNMPv1 introduced five main Protocol Data Units: (1) `GetRequest`, (2) `GetNextRequest`, (3) `SetRequest`, (4) `GetResponse`, and (5) `Trap`. `GetRequest` is used by the manager to collect the value of one or more objects managed by the agent. The manager uses `GetNextRequest` message to request a series of consecutive variables managed by the agent. `SetRequest` is used by the manager to modify the value of one or more objects in a managed device. `GetResponse` is sent by agents to respond with data to get (`GetRequest` and `GetNextRequest`) and set (`SetRequest`) requests. `Trap` is used by the agent to notify that an event has occurred or that a condition is present. SNMPv1 does not allow manager-to-manager interactions [28].

Three new PDUs were added in SNMPv2c: (1) `GetBulkRequest`, (2) `InformRequest`, and (3) `Report`. The purpose of `GetBulkRequest` is the optimization of `GetNextRequest`, allowing to request the transfer of a large amount of data and reducing the number of requests and responses. `InformRequest` is used by a manager to send management information to other remote managers. Usage and precise semantics of `Report` are not specified; therefore, any SNMP administrative framework making use of this PDU must define it. SNMPv2c improved error-handling by including expanded error codes to differentiate types of error conditions reported through a single error code in SNMPv1 [29].

B. SNMPv3/USM

The User-based Security Model (USM) provides authentication and privacy capabilities at the SNMP message level. It defines three security levels that can be summarized as follows:

- Communication without authentication and privacy (noAuthNoPriv): From a security point of view, it is comparable to the CSM used by previous versions of SNMP. Neither authentication, nor encryption for privacy capabilities, are provided.
- Communication with authentication but without privacy (authNoPriv): It provides authentication. However, encryption for privacy is not provided by this level.
- Communication with authentication and privacy (authPriv): It provides both authentication and encryption for privacy capabilities.

The USM model implements its own user and key management infrastructure, making it unpractical to be implemented [1]. It relies on the existence of pre-shared keys between two communicating SNMP engines.

C. SNMPv3/TSM

The Transport Security Model (TSM) was designed to fit into the SNMP architecture as a Security Model that utilizes the services of a secure Transport Model. The TSM model does not provide security mechanisms such as authentication and encryption itself [19]. Instead, it was implemented to work with a variety of secure transport protocols, including Secure Shell [2] (SSH), Transport Layer Security [3] (TLS), and Datagram Transport Layer Security [4] (DTLS).

1) SNMPv3/SSH: The Secure Shell (SSH) protocol [2] is used for secure remote login and other secure network services over an insecure network. It comprises of three components:

- Transport Layer Protocol: it provides server authentication, confidentiality, integrity, and compression. It operates over a TCP connection, however, other reliable data streams can be used. Public-key cryptography is used to authenticate the server to the client and to establish a secure connection, which then uses a session key and a symmetric encryption algorithm to protect the connection.
- User Authentication Protocol: it authenticates the client-side user to the server and runs over the transport layer protocol. SSH can support multiple user authentication mechanisms including, but not limited to, password authentication, public-key authentication, and keyboard-interactive authentication (which supports challenge-response authentication mechanisms). Through the Generic Security Service Application Program Interface (GSS-API), SSH can also interact with the Kerberos protocol to authenticate users.
- Connection Protocol: it multiplexes the encrypted tunnel into several logical channels. It runs over the transport layer protocol and starts once the user authentication protocol has finished.

2) SNMPv3/TLS: The Transport Layer Security (TLS) protocol [3] provides authentication, integrity, and privacy at the transport layer. The TLS Transport Model (TLSTM) for SNMP consists of a model instantiation in the transport subsystem and details the elements of procedure for sending and receiving SNMP messages over TLS. TLSTM makes use of the X.509 public key infrastructure to provide authentication.

3) SNMPv3/DTLS: The Datagram Transport Layer Security (DTLS) protocol [4] is based on the TLS protocol and provides similar security capabilities. The main difference in comparison with TLS is that DTLS provides secure communication over unreliable datagram transports (e.g., UDP).

IV. METRICS AND BENCHMARKS

Let us define the "response time" as the time required for an SNMP manager to send a request and receive the associated response from the agent. We could not find a software tool on the Internet that fulfilled our need in computing the response

time with precision. Hence, we wrote our own benchmarking tool in the C programming language, using the Net-SNMP library [7]. Basically, a request/response exchange is done several times between our benchmarking tool and the agent. The benchmarking tool takes a timestamp before and after the interchange. The difference in timestamps is divided by the number of exchanges to get the average response time. Repeating the request/response exchange several times minimizes the error on the response time, due to low-precision clocks and any other processes that could be started by the operating systems and load the devices during the benchmark execution.

The benchmark has several parameters, including the version of SNMP, the community (only for SNMPv1 and SNMPv2c), the security name, security level (noAuthNoPriv, authNoPriv, and authPriv), the authentication protocol and passphrase, the privacy protocol and passphrase (only for SNMPv3/USM), the digital certificates for the benchmarking tool and the agent (only for SNMPv3/DTLS and SNMPv3/TLS), the number of sessions (`numSessions`), the number of requests/responses per session (`sessionSize`), the transport protocol (UDP, TCP, DTLS, and TLS), the IP address of the agent, and a list of parameters related to Object Identifiers (OIDs). The latter list will depend on the petitions. For example, for `GetRequest` and `GetNextRequest` petitions, it should be the list of OIDs to be resolved into values. For `SetRequest` petitions, it should be a list of triplets (OID to be altered, its type, and its new value). Fig. 1 gives the skeleton of the benchmark for computing the response time for a `GetRequest` petition. The line numbers have been added just for reference. Line 01 gets the starting timestamp. The external for-loop controls the number of sessions (`numSessions`). For each session, the internal for-loop controls the number of requests/responses per session (`sessionSize`). Each session consists of opening the session with the agent (Line 05), repeating the creation of the request (Line 07), exchanging the request and response with the agent (Line 09), and destroying the response once processed (Line 11), before closing the session (Line 13). Finally, Line 16 gets the ending timestamp, and the results are displayed.

```
01: gettimeofday(&timerStart, (struct timezone *) 0);
02: // Get the starting timestamp
03:
04: for(int i=0; i<numSessions; i++) {
05: ss = snmp_open(&session); // Open an SNMP session
06: for(int j=0; j<sessionSize; j++) {
07: pdu = snmp_pdu_create(SNMP_MSG_GET); // Create request
08: // Add pairs of (OIDs, null) to the request
09: status = snmp_synch_response(ss, pdu, &response);
10: // Process the response
11: snmp_free_pdu(response);
12: }
13: snmp_close(ss); Close the SNMP session
14: }
15:
16: gettimeofday(&timerEnd, (struct timezone *) 0);
17: // Get the ending timestamp before showing the results
```

Fig. 1. Skeleton of the Code of the Benchmark to Compute the Response Time for a GetRequest.

V. Description of the Test Environment

The testbed of Fig. 2 was used for the experiments. It consisted of a laptop, a wireless router, and SBCs from the Raspberry Pi Foundation. The laptop and SBCs were placed 4 meters from the wireless router, with no obstacles between them. Section V.A gives more details about the different models of SBCs (RPi Zero W, RPi 3B, and RPi 3B+) that were used. The laptop had the following specifications: Microsoft Surface Book with an Intel Core i7-6600U CPU at 2.81 GHz, 16 GB of RAM, a 512 GB SSD, an NVIDIA GeForce GPU, and a Marvell AVASTAR Wireless-AC Network Adapter (dual-band wireless adapter with support to IEEE 802.11 a/b/n/g/ac). Debian amd64 10.11.0 was installed as the operating system. For the wireless network interconnection, a NETGEAR AC1200 Smart WiFi Router R6220 was employed. It had the following characteristics: an 880 MHz MediaTek processor width two radio bands (IEEE 802.11b/g/n in the 2.4 GHz band and IEEE 802.11a/n/ac in the 5 GHz band), 128 MB of flash, 128 MB of RAM, and five 10/100/1000 Mbps Ethernet ports (1 WAN and 4 LAN). In the 2.4 GHz band, the bandwidth can be set up to a maximum of 54, 145, or 300 Mbps. At the level of the 5 GHz band, a maximum of 173, 400, and 867 Mbps can be configured.

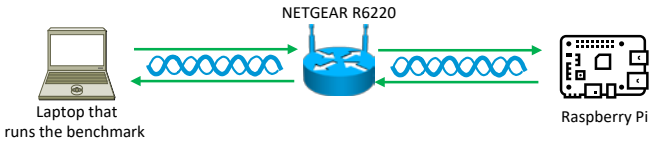

Fig. 2. Testbed for the Experiments.

A. Models of Raspberry Pi used in the Experiments

The Raspberry Pi Zero W (RPi Zero W) is based on a 32-bit Broadcom BCM2835 single-core ARM1176JZF-S SoC @ 1.0 GHz, 512 MB of RAM, one 2.4 GHz IEEE 802.11b/g/n WiFi interface, one micro USB On-The-Go port, one mini HDMI connector, and one microSD card slot. The Raspberry Pi 3 Model B (RPi 3B) is based on a 64-bit Broadcom BCM2837 quad-core Cortex-A53 SoC @ 1.2 GHz, 1 GB of RAM, one 10/100 Mbps Ethernet interface, one 2.4 GHz IEEE 802.11b/g/n WiFi interface, four USB 2.0 ports, one full-size HDMI connector, and one microSD card slot. The Raspberry Pi 3 Model B+ (RPi 3B+) is based on a 64-bit Broadcom BCM2837B0 quad-core Cortex-A53 SoC @ 1.4 GHz, 1 GB of RAM, one Gigabit Ethernet interface over USB 2.0 (maximum throughput 300 Mbps), one dual-band 2.4 GHz and 5 GHz IEEE 802.11a/b/g/n/ac WiFi interface, four USB 2.0 ports, one full-size HDMI connector, and one microSD card slot.

B. Operating Systems for Raspberry Pi

Many operating systems are available for Raspberry Pi (e.g., Raspberry Pi OS, Debian, Ubuntu, RaspBSD, Kali Linux, OpenSUSE, RetroPie, LibreELEC, RISC OS). We opted for Raspberry Pi OS (32-bit), released in May 2021, which is the continuity of Raspbian (one of the most accepted OS for Raspberry Pi, worldwide). The Raspberry Pi Foundation offers three versions of this operating system that are compatible with all Raspberry Pi models: (1) Raspberry Pi OS Lite, (2) Raspberry Pi OS with Desktop, and (3) Raspberry Pi OS with Desktop and Recommended Software. The “Lite” version does

not have a GUI, and therefore it is faster since it does not have the full overload of a desktop environment. It is totally based on the command-line interface (terminal) and consists of 483 packages. The "Desktop" version has all the features of the "Lite" version, but also includes software such as Openbox as the window manager and LXDE (Lightweight X11 Desktop Environment) as the desktop environment. It consists of 1384 packages. The "Desktop and Recommended Software" version has all the "Desktop" version features, but also includes additional software such as LibreOffice, Firebird, Apache Ant, BlueJ, Greenfoot, OpenJDK Java Runtime Environment, OpenJDK Java Development Kit, Node.js, and Ruby. It consists of 2021 packages. We chose the "Lite" version since an SBC that is running an SNMP agent will most likely be headless, without the need of a GUI.

The Raspberry Pi Foundation also has a 64-bit version of its operating system that can be run only in 64-bit based hardware like the RPi 3B, RPi 3B+, RPi 4B, and RPi 400. That is, it is not suitable for the RPi Zero W. It is worth mentioning that it is still in the beta stage, and not directly advertised on the website of the Raspberry Pi Foundation, since they are still working on fixing issues that does not have the 32-bit version.

The performance of a Raspberry Pi will be noticeably affected by its microSD card. In the three SBCs, the original microSD card was replaced by a 64 GB SanDisk Extreme microSDXC UHS-I Memory Card (SDSQXA2-064G-GN6MA). It is considered as one of the fastest microSD cards of the market, with up to 160 MB/s and 60 MB/s for the reading and writing speeds, respectively.

C. Compiling Net-SNMP

Net-SNMP [7] is a widely used open-source, comprehensive implementation of the SNMP protocol. It has support for all the versions of SNMP and consists of an agent (`snmpd`) and several client applications (`snmpget`, `snmpgetnext`, `snmpset`, `snmpbulkget`, `snmpwalk`, etc). Precompiled packets for Net-SNMP v5.7.3 are available in the repositories of Raspberry Pi OS. However, at the level of SNMPv3, they were compiled to support the USM model, but not the TSM model. Hence, a newer version of Net-SNMP (v5.8) was compiled and installed in all the Raspberry Pis. To this end, the commands of Fig. 3 were executed. The required libraries were first installed from the repositories. At the configuration level, the security models (both USM and TSM) and the transport protocols (UDP, TCP, UDPIPv6, TCPIPv6, DTLSUDP, TLSTCP, and SSH) were specified.

Table I shows the necessary time for each phase of the compilation and installation process (configuration, compilation, and installation) for the different Raspberry Pis that were used in this work. These results can be beneficial, since they shed light on the power of each SBC.

```
apt-get install libssl-dev libperl-dev libssh2-1-dev
tar zxvf net-snmp-5.8.tar.gz
cd net-snmp-5.8
./configure --with-security-modules=usm,tsm \
 --with-transports=UDP,TCP,UDPIPv6,TCPIPv6,DTLSUDP,TLSTCP,SSH
make
make install
```

Fig. 3. Commands to Compile and Install Net-SNMP.

TABLE I. COMPILATION TIMES OF NET-SNMP

Command	RPi Zero W	RPi 3B	RPi 3B+
./configure	15m42s	4m31s	4m3s
make	62m26s	14m14s	12m28s
make install	4m8s	1m18s	1m7s

It is worth clarifying that the recent versions of Net-SNMP [7] have experimental support for SNMPv3/SSH. Despite many efforts, this research team could not successfully install and use it. There is little documentation on setting the environment of SNMPv3/SSH. Hence, we did not report results related to this specific security model in this paper.

VI. PERFORMANCE RESULTS AND ANALYSIS

Here, we describe the common parameters selected for all our experiments:

- We configured the radios of the equipment in the 2.4 GHz band. The wireless router was set up to a maximum of 54 Mbps.
- Recent versions of SNMP can use UDP or TCP as the transport protocol. SNMP was initially designed for UDP, and will most likely be used with UDP since most SNMP agents are developed to use this protocol (it requires less computing power than TCP). Hence, otherwise stated, our experiments were done using UDP as the transport protocol.
- SNMPv3/USM has two authentication protocols (MD5 and SHA-1) and two privacy protocols (DES and AES). Unless otherwise specified, in our experiments with SNMPv3/USM, we selected SHA-1 as the authentication protocol and AES as the privacy protocol, when used. SHA-1 was preferred due to the attack on MD5 [30]. AES was selected since DES has a relatively short 56-bit key that is easily breakable with modern computers [31][32]. In January 1999, distributed.net and the Electronic Frontier Foundation were the first to collaborate and publicly broke a DES key in less than 23 hours.
- The OIDs retrieved and modified in our experiments were strings of 32 characters.
- For the experiments with SNMPv3/DTLS, self-signed certificates were generated, using the RSA algorithm with 2048-bit keys.

They are many parameters that can be varied to analyze their effects on the performance of SNMP. In this study, we considered parameters such as the type of requests, the number of objects involved per request, the security levels of SNMPv3/USM, the authentication and privacy protocols of SNMPv3/USM, the transport protocols, and the versions and security models of SNMP. Also, to get consistent results, it is worth mentioning that we repeated each experiment at least fifteen times, and the results presented in the study is an avarege of them.

A. Type of Requests Variation

This experiment aims to study how varying the type of requests can affect the performance of SNMP on a Raspberry Pi. The PDUs available in SNMP are version-specific. However, `GetRequest`, `GetNextRequest`, and `SetRequest` are present in all the versions, and therefore are the most commonly used requests. In this first experiment, we compared the response time of these three requests for SNMPv1, SNMPv2c, SNMPv3/USM, and SNMPv3/DTLS. The experiment is focused on sessions with a single request/response exchange. Table II shows the results for the RPi Zero W, RPi 3B, and RPi 3B+ as an agent. The differences between `GetRequest` and `GetNextRequest` petitions are not noticeable. However, the response time for a `SetRequest` is much longer, due to the reading and writing speed in the microSD card (maximum 160 MB/s and 60 MB/s for reading and writing speed, respectively).

At the level of the SNMP versions, SNMPv1 and SNMPv2c have very similar performances. SNMPv3/USM and SNMPv3/DTLS have much longer response times due to the overhead of the authentication and privacy mechanisms. It is also worth mentioning that in this experiment, SNMPv3/USM outperforms SNMPv3/DTLS, with minor differences.

In all the subsequent experiments, we focused on GetRequest petitions, since they are the most common petitions, and the majority of deployments of SNMP are focused on monitoring (not configuring), which requires massive GetRequest and GetNextRequest petitions, rather than SetRequest.

B. Number of OIDs Variation

In this experiment, the impact of the number of OIDs in the response time of a `GetRequest` petition is studied, and it was varied from 1 to 32. The experiment is focused on sessions with a single request/response exchange.

Fig. 4 and Fig. 5 depict the results obtained for SNMPv1 and SNMPv2c, respectively. Our study seems to indicate that both have a very similar performance.

TABLE II. RESPONSE TIME OF DIFFERENT REQUESTS (MILLISECONDS)

Type of Request	Version	RPi Zero W	RPi 3B	RPi 3B+
`GetRequest`	v1	2.97	2.02	1.71
	v2c	2.99	2.01	1.73
	v3/USM	3.51	2.24	2.15
	v3/DTLS	3.95	2.44	2.36
`GetNextRequest`	v1	2.98	2.08	1.74
	v2c	2.96	2.05	1.72
	v3/USM	3.53	2.31	2.20
	v3/DTLS	4.02	2.47	2.33
`SetRequest`	v1	151.27	127.44	122.35
	v2c	151.35	127.50	122.37
	v3/USM	155.32	131.74	126.92
	v3/DTLS	157.21	135.87	131.56

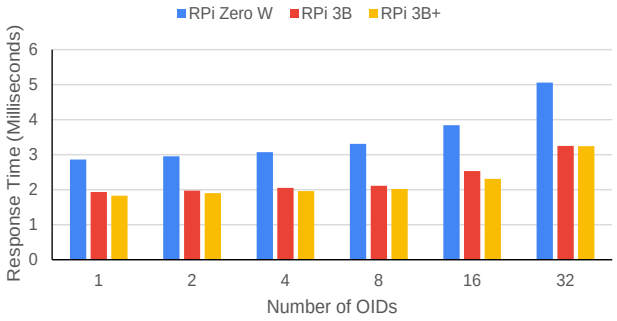

Fig. 4. Response Time for a GetRequest when Varying the Number of OIDs for SNMPv1.

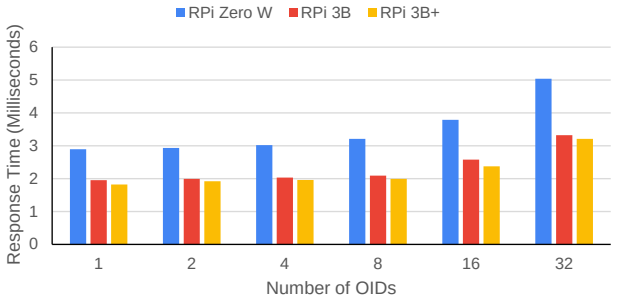

Fig. 5. Response Time for a GetRequest when Varying the Number of OIDs for SNMPv2c.

Fig. 6 and Fig. 7 show the results obtained for SNMPv3/USM with authPriv (SHA-1 and AES) and SNMPv3/DTLS, respectively. The response time for SNMPv3/USM is slightly longer than for SNMPv1 and SNMPv2c. SNMPv3/DTLS has the longest response time.

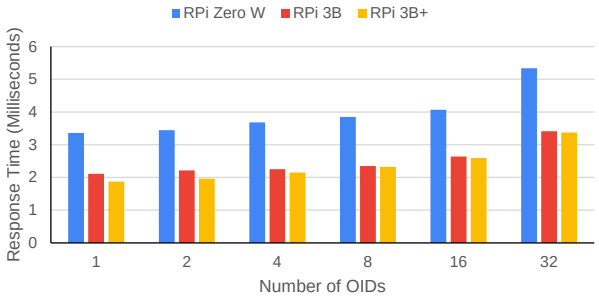

Fig. 6. Response Time for a GetRequest when Varying the Number of OIDs for SNMPv3/USM.

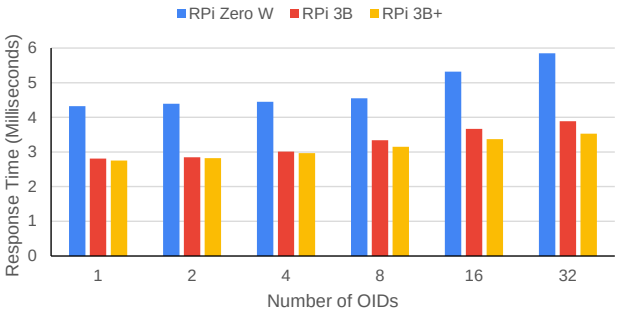

Fig. 7. Response Time for a GetRequest when Varying the Number of OIDs for SNMPv3/DTLS.

The tendency of this experiment indicates that the response time for a `GetRequest` will be linearly proportional to the number of OIDs. It is also noticeable that both the RPi 3B and the RPi 3B+ have similar results, which are much better than the RPi Zero W.

C. Security Level Variation for SNMPv3/USM using UDP and TCP as Transport Protocols for the RPi Zero W

The objective of this experiment is to analyze the impact of the security levels (noAuthNoPriv, authNoPriv, and authPriv) when using SNMPv3/USM on an RPi Zero W. The experiment is also aimed at understanding how the transport protocol (UDP or TCP) can affect the performance. To simplify the notation, let us abbreviate noAuthNoPriv as "nn", authNoPriv as "an", and authPriv as "ap".

Fig. 8 depicts the total response time for our experiments for different numbers of requests/responses in a session (from 50 to 400 requests/responses). For each size of the session, six total response times are reported: (1) noAuthNoPriv with UDP, (2) noAuthNoPriv with TCP, (3) authNoPriv with UDP, (4) authNoPriv with TCP, (5) authPriv with UDP, and (6) authPriv with TCP. We selected SHA-1 and AES as the authentication and privacy protocols, respectively.

As indicated by our experiments, TCP has a slightly longer response time, but the differences with UDP are not significant. The variations due to the different privacy levels are more noticeable. As expected, noAuthNoPriv is the shortest response time, while authPriv is the longest.

D. Authentication and Privacy Protocols Variation for SNMPv3/USM using UDP as the Transport Protocol for the RPi Zero W

This experiment aims to assess the impact of the authentication protocols (MD5 and SHA-1) and the privacy protocols (DES and AES) when using SNMPv3/USM on an RPi Zero W.

Fig. 9 depicts the total response time of our experiments for different numbers of requests/responses in a session (from 50 to 400 requests/responses). For each size of the session, seven total response times are reported: (1) noAuthNoPriv, (2) authNoPriv with MD5, (3) authNoPriv with SHA-1, (4) authPriv with MD5 and DES, (5) authPriv with MD5 and AES, (6) authPriv with SHA-1 and DES, and (7) authPriv with SHA-1 and AES.

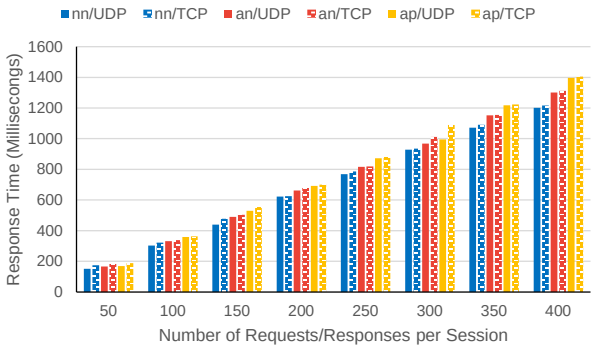

Fig. 8. Total Response Time for a Session of GetRequest for SNMPv3/USM when Varying the Security Level and the Transport Protocol.

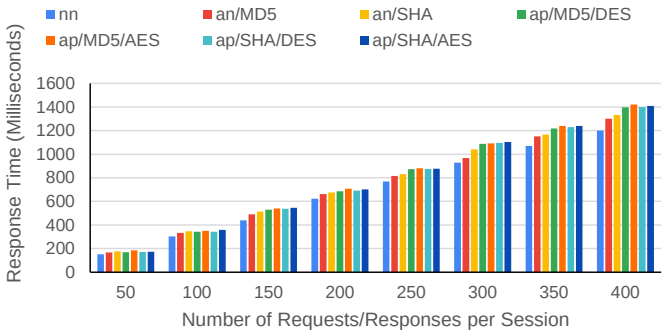

Fig. 9. Total Response Time for a Session of GetRequest for SNMPv3/USM when Varying the Authentication and Privacy Protocols.

Our results seem to indicate that MD5 is faster than SHA-1 as an authentication protocol. However, it is worth reminding that MD5 is now considered insecure [30]. Also, at the level of the privacy protocol, DES appears to be faster.

E. SNMPv3/USM vs SNMPv3/DTLS

In this experiment, we investigate the performance of SNMPv3/USM (SHA-1 and AES) vs. SNMPv3/DTLS on the RPi Zero W, RPi 3B, and RPi 3B+.

Fig. 10 depicts the total response time of our experiments for different numbers of requests/responses in a session (from 50 to 400 requests/responses). For each size of the session, six total response times are reported: (1) SNMPv3/USM (SHA-1 and AES) for RPi Zero W, (2) SNMPv3/DTLS for RPi Zero W, (3) SNMPv3/USM (SHA-1 and AES) for RPi 3B, (4) SNMPv3/DTLS for RPi 3B, (5) SNMPv3/USM (SHA-1 and AES) for RPi 3B+, and (6) SNMPv3/DTLS for RPi 3B+.

The best results are obtained by the RPi 3B+, while the worst correspond to the RPi Zero W. Also, this experiment confirmed that SNMPv3/USM has a better performance than SNMPv3/DTLS, as already mentioned in Section VI.A.

Notice that we also did experiments with SNMPv3/TLS. However, the obtained results were not stable at all, and we had significant variations of the response time from one test to another. Hence, we decided not to report them in this paper.

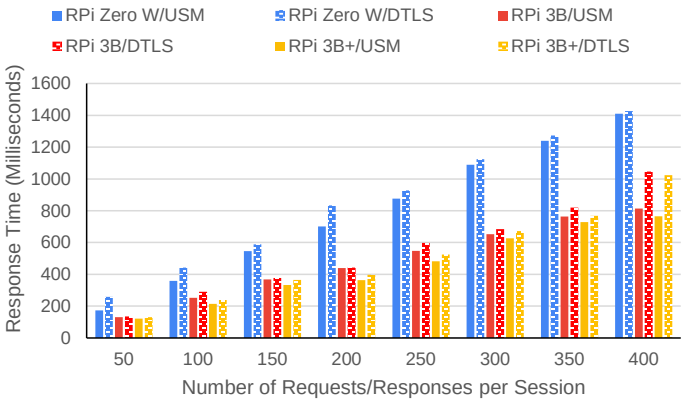

Fig. 10. Total Response Time for a Session of GetRequest for SNMPv3/USM and SNMPv3/DTLS.

F. Retrieving the Interface Table with snmpwalk when Varying the Number of Interfaces

As mentioned previously, Net-SNMP [7] has several client applications (snmpget, snmpgetnext, snmpset, snmpbulkget, snmpwalk, etc). In this experiment, we investigated the performance of snmpwalk by retrieving the interface table (ifTable [23]), when varying the numbers of interfaces, for SNMPv1, SNMPv3/USM (SHA-1 and AES), and SNMPv3/DTLS, on the RPi Zero W, RPi 3B, and RPi 3B+. snmpwalk uses GetNextRequest requests to query an agent for a portion of the object identifier space (e.g., a table). All objects in the subtree below a given OID are queried and their values are presented to the user. We varied the number of interfaces from 2 to 64, by creating additional dummy interfaces on the SBCs as specified in Fig. 11. The output of the application was discarded by redirecting it to /dev/null.

```
modprobe dummy
for i in $(seq $startValue $endValue)
do
 echo "Creating interface eth${i} with address 10.0.0.${i}/32"
 ip link add eth${i} type dummy
 ip address add 10.0.0.${i}/32 dev eth${i}
 ip link set up dev eth${i}
done
```

Fig. 11. Creation of Dummy Interfaces in the Agents.

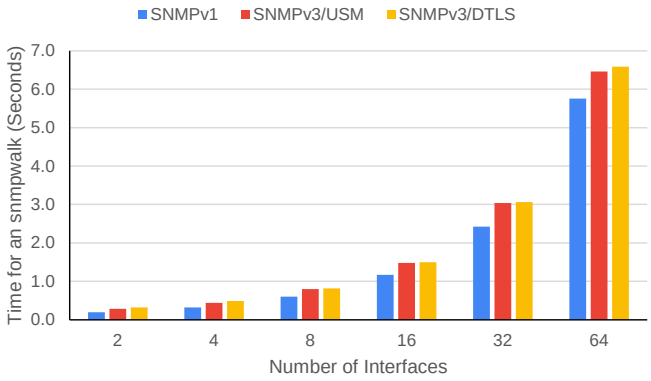

Fig. 12. Time to Retrieve the ifTable using Snmpwalk when Varying the Number of Interfaces for the RPi Zero W.

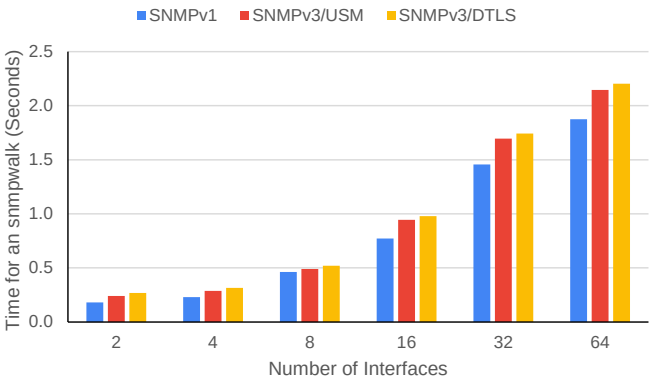

Fig. 13. Time to Retrieve the ifTable using Snmpwalk when Varying the Number of Interfaces for the RPi 3B.

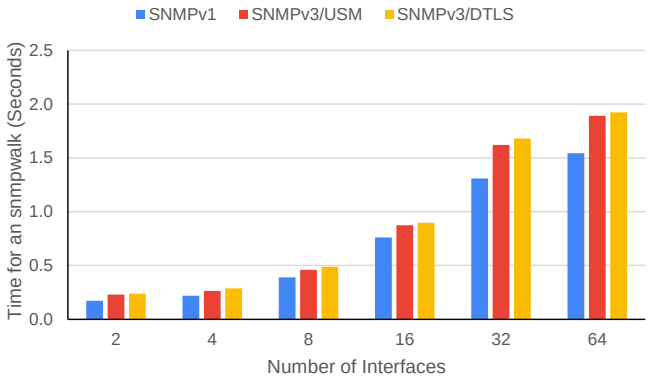

Fig. 14. Time to Retrieve the ifTable using Snmpwalk when Varying the Number of Interfaces for the RPi 3B+.

Fig. 12, 13, and 14 depict the time to retrieve the interface table (`ifTable`) through the `snmpwalk` application for the RPi Zero W, RPi 3B, and RPi 3B+, respectively. For small numbers of interfaces, SNMPv1 has results that are similar to the ones of SNMPv3/USM (SHA-1 and AES) and SNMPv3/DTLS. However, as the number of interfaces increases, the processing time becomes predominant over the transmission time, resulting in bigger differences between SNMPv1 and the other two versions of SNMP.

VII. Conclusion and Future Work

Our experiments seem to indicate that SNMPv1 and SNMPv2c have similar performances. The assessment results of SNMPv3/USM and SNMPv3/DTLS are close to each other, with a slight advantage for the former. At the level of the SBCs, the RPi 3B and RPi 3B+ performed mostly equally, with the latter slightly outperforming the former. We found significant differences in the response time of `GetRequest` and `SetRequest`. We believe that these differences are due to the reading and writing access to the microSD cards (up to 160 MB/s and 60 MB/s for the reading and writing speeds, respectively).

Unfortunately, and despite all our efforts, we could not succeed in using SNMPv3/SSH with Net-SNMP. Also, SNMPv3/TLS gave inconsistent results from test to test, so we decided not to report them in this study.

As future work, we plan to evaluate SNMP, RESTCONF [17], and NETCONF [17] as management solutions in different scenarios. Also, with the growing adoption of IPv6, we are interested in analyzing the influence of the network protocol (i.e., IPv4 and IPv6) over the SNMP performance.

Acknowledgment

We are grateful to “Faculty Commons” and the “College of Science & Mathematics” at Jacksonville State University for partially funding this project.

References

[1] J. Schönwälder and V. Marinov, “On the Impact of Security Protocols on the Performance of SNMP,” IEEE Transactions on Network and Service Management, vol. 8, no. 1, March 2011, pp. 52–64.

[2] M. Lucas, SSH Mastery: OpenSSH, PuTTY, Tunnels and Keys, Tilted Windmill Press; 2nd edition, February 2018.

[3] E. Rescorla, The Transport Layer Security (TLS) Protocol Version 1.3, RFC 8446, August 2018.

[4] E. Rescorla, H. Tschofenig, and N. Modadugu, The Datagram Transport Layer Security (DTLS) Protocol Version 1.3, Draft IETF, April 2021.

[5] S. Monk, Programming the Raspberry Pi: Getting Started with Python, McGraw-Hill Education TAB, 3rd edition, June 2021.

[6] L. Clark, Raspberry Pi 4: The Ultimate Step-by-Step Guide to Using Raspbian to Create Incredible Projects and Expand Your Programming Skills with the Latest Version of Raspberry Pi, independently published, February 2021.

[7] Net-SNMP Home Page, http://www.net-snmp.org.

[8] L. Andrey, O. Festor, A. Lahmadi, A. Pras, and J. Schönwälder, "Survey of SNMP Performance Analysis Studies," International Journal of Network Management, vol. 19, 2009, pp. 527–548.

[9] F. Hidalgo and E. Gamess, "Integrating Android Devices into Network Management Systems based on SNMP," International Journal of Advanced Computer Science and Applications, vol. 5, no. 5, 2014, pp. 1–8.

[10] A. Corrente and L. Tura, Security Performance Analysis of SNMPv3 with Respect to SNMPv2c, in Proceedings of the 2004 IEEE/IFIP Network Operations and Management Symposium (NOMS 2004), Seoul, South Korea, April 2004.

[11] F. Duarte and A. Loureiro, "Performance Evaluation and Scalability Analysis of SNMPv3 with Superimposition in a Mobile Environment," Concurrent Engineering Research and Applications, vol. 9, no. 2, June 2001, pp 139–145.

[12] A. Pras, T. Drevers, R. van de Meent, and D. Quartel, "Comparing the Performance of SNMP and Web Services-Based Management," IEEE Transactions on Network and Service Management, vol. 1, no. 2, December 2004.

[13] P. Santos, R. Esteves, and L. Granville, Evaluating SNMP, NETCONF, and RESTful Web Services for Router Virtualization Management, in Proceedings of the 2015 IFIP/IEEE International Symposium on Integrated Network Management (IM 2015), Ottawa, ON, Canada, May 2015.

[14] M. Słabicki and K. Grochla, Performance Evaluation of SNMP, NETCONF and CWMP Management Protocols in Wireless Network, in Proceedings of the 4th International Conference on Electronics, Communications and Networks, Beijing, China, December 2014.

[15] M. Słabicki and K. Grochla, Performance Evaluation of CoAP, SNMP and NETCONF Protocols in Fog Computing Architecture, in Proceedings of the 2016 IEEE/IFIP Network Operations and Management Symposium (NOMS 2016), Istanbul, Turkey, April 2016.

[16] Q. Gu and A. Marshall, Network Management Performance Analysis and Scalability Tests: SNMP vs CORBA, in Proceedings of the 2004 IEEE/IFIP Network Operations and Management Symposium (NOMS 2004), Seoul, South Korea, April 2004.

[17] B. Claise, J. Clarke, and J. Lindblad, Network Programmability with YANG: The Structure of Network Automation with YANG, NETCONF, RESTCONF, and gNMI, Addison-Wesley Professional, 1st edition, May 2019.

[18] X. Du, M. Shayman, and M. Rozenblit, Implementation and Performance Analysis of SNMP on a TLS/TCP Base, in Proceedings of the 2001 IEEE/IFIP International Symposium on Integrated Network Management, Seattle, WA, USA, May 2001.

[19] D. Harrington and W. Hardaker, Transport Security Model for the Simple Network Management Protocol (SNMP), RFC 5591, June 2009.

[20] D. Harrington, J. Salowey, and W. Hardaker, Secure Shell Transport Model for the Simple Network Management Protocol (SNMP), RFC 5592, June 2009.

[21] W. Hardaker, Transport Layer Security (TLS) Transport Model for the Simple Network Management Protocol (SNMP), RFC 6353, July 2011.

[22] V. Marinov and J. Schönwälder, "Performance Analysis of SNMP over SSH," Lecture Notes in Computer Science, vol. 4269. Springer, Berlin, Heidelberg, 2006. https://doi.org/10.1007/11907466_3

[23] K. McCloghrie and M. Rose, Management Information Base for Network Management of TCP/IP-based Internets: MIB-II, RFC 1112, March 1991.

[24] J. Case, M. Fedor, M. Schoffstall, and J. Davin, A Simple Network Management Protocol, RFC 1067, August 1988.

[25] M. Julian, Practical Monitoring: Effective Strategies for the Real World. O'Reilly, 1st edition, November 2017.

[26] D. Harrington, R. Presuhn, and B. Wijnen, An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks, RFC 3411, December 2002.

[27] U. Blumenthal and B. Wijnen, User-based Security Model (USM) for version 3 of the Simple Network Management Protocol (SNMPv3), RFC 3414, December 2002.

[28] R. Presuhn, J. Case, K. McCloghrie, M. Rose, and S. Waldbusser, Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP), RFC 3416, December 2002.

[29] J. Case, K. McCloghrie, M. Rose, and S. Waldbusser, Protocol Operations for version 2 of the Simple Network Management Protocol (SNMPv2), RFC 1448, April 1993.

[30] T. Xie, F. Liu, and D. Feng, "Fast Collision Attack on MD5," IACR Cryptology ePrint Archive, vol. 2013, 2013.

[31] M. Curtin, Brute Force: Cracking the Data Encryption Standard, Copernicus; 2005th edition, February 2005.

[32] E. Biham and A. Biryukov, "An Improvement of Davies' Attack on DES," Joumol of Cryptology, vol. 10, June 1997, pp. 195-205.