UNIVERSIDADE DA BEIRA INTERIOR
Faculdade de Engenharia
Departamento de Informática

Monitoring platform for the UBI network infrastructure

Richard Albert Correia Guise

Submitted to the University of Beira Interior in candidature for the Degree of Master of Science in Informatics Engineering

Supervised by Prof. Nuno M. Garcia

Department of Informatics
University of Beira Interior
Covilhã, Portugal
http://www.di.ubi.pt

Acknowledgements

I would like to express my deepest gratitude to my friend, colleague and supervisor, Prof. Nuno M. Garcia for his guidance, useful critiques and invaluable expertise in the formulating of this dissertation.

I am particularly grateful for the assistance given by my friend and colleague Hugo Veiga and for his valuable and constructive suggestions during the planning and development phases. His willingness to give his time so generously has been very much appreciated since day one in the networking department.

I would also like to extend my gratitude to the director of SIUBI, Eng. Paulo Gomes, who provided me with the opportunity to pursue a new career in networking.

A special thank you to Prof. Mário Freire, for giving me the opportunity to reenroll in this course and resume my studies.

In addition, I am grateful for my parents and the opportunities they have provided during my life.

Finally and most importantly, I wish to thank my loving wife and daughters, for all their encouragement, support and especially patience throughout my work and studies.

Abstract

Network monitoring is a crucial IT process, which consists of monitoring network devices such as routers, switches, firewalls and servers for performance and fault issues. A good functioning network if vital for an organization, but unfortunately, network outages and performance issues are a part of every organization's network. Faults, being hardware or human originated, may appear at any time and can give rise to sometimes critical situations. For this reason, network devices should be monitored continuously in a proactive way to prevent these network failures and downtimes. Identifying traffic bottlenecks, faulty components, low performance and other types of issues in an early stage minimizes or even eliminates bigger problems that can occur later on. Efficient proactive monitoring can help prevent network outage and should be implemented by every network administrator. Adopting a secure, low bandwidth consumption and compatible protocol is a good practice when implementing a monitoring solution. One such protocol is the Simple Network Management Protocol (SNMP) and provides a message format for communication between the SNMP managers and agents; it is also supported by most of the present day network devices and servers. The main goal of research described in this dissertation is the study of the various existing freeware SNMP monitoring platforms in the market today and the implementation of the one best suited for the university's network. The solution would have to be compatible with the university's multivendor device network and be scalable enough to permit future growth. It should also have a good alerting system to provide a pro-active approach to resolving issues. Implementation, evaluation and conclusions of the best suited monitoring solution are presented during the course of this study.

Keywords

Network Monitoring, Zabbix, Simple Network Management Protocol

Contents

List of Figures

List of Tables

Acronyms

API	Application Programming Interface
CENTOS	Community ENTerprise Operating System
CLI	Command Line Interface
DVD	Digital Versatile Disc
GUI	Graphical User Interface
HTTP	HyperText Transfer Protocol
IP	Internet Protocol
ISO	International Organization for Standardization
KDE	Kool desktop environment
LACP	Link Aggregation Control Protocol
MIB	Management Information Base
MYSQL	My Structured Query Language.
NMS	Network Management System.
OID	Object Identifiers
OS	Operating System
PC	Personal Computer
PDU	Protocol Data Unit
PHP	PHP: Hypertext Preprocessor
PSU	Power Supply Unit
SI	*Serviços de Informática da Universidade da Beira Interior*
SMS	Short Message Service
SNMP	Simple Network Management Protocol
SSD	Solid-State Drive

SQL	Structured Query Language
UEFI	Unified Extensible Firmware Interface
USB	Universal Serial Bus
XML	eXtensible Markup Language

Chapter 1

1. Introduction

A network administrator is responsible for keeping an organization's computer network up-to-date and operating as intended. The duties of a network administrator can vary from generalist work, which covers everything from infrastructure setup to security system management, to other tasks that have a more specific focus. To carry out these duties, one should possess a set of tools that are essential for keeping an eye on the network at all times by means of monitoring and alarm functionalities.

Adopting a secure, low bandwidth consumption and compatible protocol is a good practice when implementing a monitoring solution in the organization's network. One such protocol is the Simple Network Management Protocol (SNMP), which is used for equipment monitoring and management. It provides a way for different devices on a network to share information with one another. The SNMP protocol is also integrated in a wide variety of devices, applications and solutions.

There is a wide array of SNMP monitoring tools in the market today, ranging from proprietary paid platforms all the way to open source freeware options. This study will focus on the freeware networking monitoring solutions that are the most suited for the University of Beira Interior network.

1.1. Objective

The main goal of this work is the study of the various existing freeware SNMP monitoring platforms in the market today, the identification of the one best suited for the university's network, and its implementation and test. The device monitoring should be accomplished by means of the SNMP protocol, due to it being platform independent and having a wide range of supported

hardware such as such as switches, routers, and gateways, as well as on endpoint devices such as printers and servers.

1.2. Motivation

The UBI computer network has experienced a significant growth over the years, in terms of number of connected hosts and diversity of network equipment. Due to the network's topology and multivendor environment, the implementation of a new SNMP monitoring solution was long overdue to assist the network administrator's work in a pro-active way. This study will assist in choosing the right monitoring system for the university's network topology, keeping in mind the budget limitation.

1.3. Contribution

This work focuses on the study, analysis and implementation of a much-needed network monitoring solution for the University of Beira Interior network. The main contribution of this work is to define a monitoring solution that can be integrated in the university's network, and eventually, be put in the production environment.

1.4. Organization

This dissertation is organized and structured in five chapters that demonstrate the process used to accomplish the work's objectives. The chapters are the following:

- **Chapter 1:** consists of a general introduction of the dissertation theme presenting the objectives, motivations and contributions of the project's implementation.
- **Chapter 2**: focuses on the study of the concepts and operation of the SNMP protocol. Analysis of the existing SNMP monitoring freeware software available in the market today and the reasons for choosing a particular one.

- **Chapter 3:** presents a brief overview of the UBI network topology and explains the steps taken in implementing the monitoring solution chosen in Chapter 2.
- **Chapter 4:** presents an analysis and evaluation of the results obtained from the implemented network monitoring solution.
- **Chapter 5:** presents the conclusions of this study and proposals for future work.

Chapter 2

2. State of the Art

Network monitoring is a crucial IT process, which consists of monitoring network devices such as routers, switches, firewalls and servers for performance and fault issues. Unfortunately, network outages and performance issues are a part of every organization's network. Outages are not uncommon and hours typically pass before an issue is reported and resolved.

Network devices should be monitored continuously in a proactive way to prevent network failures and downtimes. Identifying traffic bottlenecks, faulty components, low performance and other types of issues in an early stage minimizes or even eliminates bigger problems that can occur later on. Efficient proactive monitoring can help prevent network outage and should be implemented by every network administrator.

There are various network monitoring solutions in the market today and choosing the right tool that best suits the requirements of an organization can be a daunting task. These software solutions come in a wide range of options, from proprietary to Internet Standard protocol based. The Simple Network Management Protocol (SNMP) is one of these options and is implemented in a wide range of network devices such as routers, switches, firewalls and also on endpoint devices such as servers and printers. Another advantage in choosing a SNMP solution-based platform is that it is an open standard protocol, which is easy to implement on large networks. SNMP is widely supported by practically all major vendors of internetwork devices, making it very easy to implement. Because of its open based protocol nature, SNMP reduces developing costs, while guaranteeing future updates and support from the industry.

This chapter presents some basic concepts of network monitoring; the elements of operation of SNMP and proceeds to discuss some of the most used free monitoring tools.

2.1. Basics of Network Monitoring

There are some important aspects to consider when implementing a network monitoring solution [1]:

- What to monitor
- When to monitor
- What protocol should be used for monitoring
- What thresholds should be defined for monitoring.

The first aspect is to decide what devices are to be monitored. Identifying crucial networking devices is important to eliminate or at least minimize outages by proactively monitor the relevant devices, like routers, switches, servers and firewalls that are critical and can affect network performance if faulty.

The second is to define monitoring intervals at which the network devices and its related metrics are polled to assess their performance and availability status. If every device is monitored at the minimal interval, this will cause unnecessary load on the devices and bandwidth consumption on the network. Therefore, the polling interval will depend on the type of network device or parameter being monitored. For example, the availability status of a device can have an interval of one minute, the memory utilization parameter can be five minutes, while the operating system version could be one day.

Adopting a secure and low bandwidth consumption protocol is a good practice when implementing a monitoring solution. The protocol should be compatible with the organizations network devices. SNMP is supported by most of the network devices and servers by means of an SNMP agent, which

just needs to be activated and configured to communicate with the monitoring solution.

Thresholds are an important part in monitoring because they identify possible problems and help in proactively fixing the concerning issues. Thresholds should be configured based on type of device, resource and need. Once a threshold is set, alerts can be raised before the device reaches a critical state and inform the network administrator of the condition. For example, a threshold can be set for the maximum percentage of memory utilization on a switch, *i.e.*, if the switch has a memory usage of over 95% then an alert can be fired to the monitoring platform.

Besides these aspects, the monitoring solution should also provide a presentable dashboard for viewing the monitoring information. The network administrator should be able to easily and clearly view the current status of the network, critical metrics from the network devices and real-time performance graphs for quickly troubleshooting problems as they arise.

2.2. The Simple Network Management Protocol

The Simple Network Management Protocol (SNMP) was developed to allow network administrators to monitor and manage network devices such as routers, switches, security appliances and servers on an IP network. It provides a way for the network administrator to easily find and solve network related issues.

SNMP is managed by the Internet Engineering Taskforce and fits into the application layer of the TCP/IP protocol stack. SNMP has gone through three versions over the years: SNMPv1, SNMPv2c and SNMPv3. SNMPv2c is the most widely used variant of version 2 due to its “community-based” authentication. SNMPv3 is the most recent version with added security features, such as user-based authentication, message integrity and the ability to use secure transport layer protocols, such as SSH and TLS in order to provide encryption protection to messages.

SNMP provides a message format for communication between managers and agents. An SNMP network includes the following three elements:

- SNMP managed devices, such as routers, switches, firewalls, printers and servers
- SNMP software agents that monitor the status of the devices
- SNMP manager, usually a network management system that requests, compiles, and stores device status information.

The controlling element of SNMP is the network manager, which is part of a network management system (NMS). The SNMP manager runs SNMP management software, which can be installed on any computer on the network. Each device on the network has a SNMP software agent installed and the communication of SNMP takes place between the manager and the agents. The SNMP manager can collect information from the SNMP agents whom respond to requests from the manager. The information gathering is done by the `get` action, but there is also a `set` action, which can change the device agent configuration. In addition, SNMP agents can forward information directly to a network manager using `traps`. Figure 2.1 illustrates some SNMP elements and the flow of those messages.

An important part of SNMP is the Management Information Base (MIB), which resides alongside the agent on the client device. The MIB is a communications framework that defines the data format exchanged between the network manager and the agent. MIBs store data about the device and operational statistics and are meant to be available to authenticated remote users.

The MIB organizes variables hierarchically and these variables enable the management software to monitor and control the device. This hierarchy has two forms of notation: number and variable names. Either naming system can be used for interaction between the manager and the agent. The MIB defines each variable as an object ID (OID), which uniquely identify managed objects in the MIB hierarchy. The MIB organizes the OIDs based on RFC

standards into a hierarchy of OIDs, usually shown as a tree. The MIB tree for any given device includes some branches with variables common to numerous networking devices and some branches with variables specific to that device or vendor. The OIDs at each point in the hierarchy depend on inheritance, for example, a root address would be 1 and then all nodes under that point in the tree would also include 1, *e.g.* 1.1, 1.2, 1.3, etc. Figure 2.2 illustrates part of an OID hierarchy tree of a Cisco MIB.

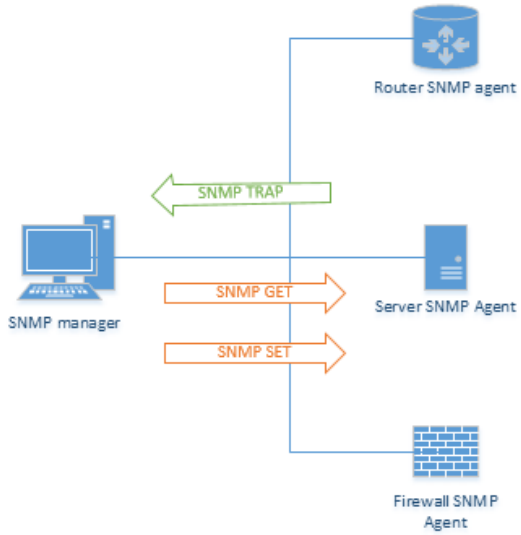


Figure 2.1 – Example of some SNMP elements.

The SNMP agent is responsible for providing access to the local device MIB. The SNMP manager polls the agents and queries the MIB for SNMP agents on UDP port 161. SNMP agents send any SNMP traps to the SNMP manager on UDP port 162. The SNMP agent monitors the device on which it resides and collects and stores information about the device and its operation. It creates a local MIB, maintaining the status of each category in the database, ready to respond to requests for information from the network manager. The agent does not automatically communicate with the NMS, but only responds to managers configured to access it. This feature strengthens the security of SNMP because the NMS has to authenticate with the agent before it responds.

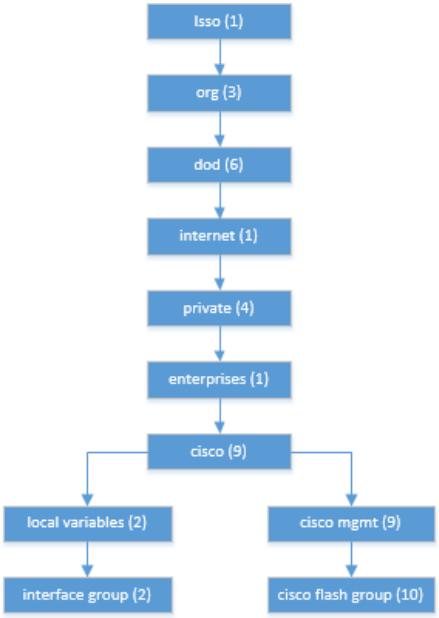


Figure 2.2 - Example of a Cisco OID Tree.

The Simple Network Management Protocol includes the definition of seven network message types, also known as Protocol Data Units (PDU). These are:

- `get-request` -Used by the SNMP manager to query the device for data;
- `getnext-request` - Retrieves a value from a variable within a table through a sequential search. It would start at a specific point in the MIB hierarchy with a `Get` request, and then continue through point by point with a series of `GetNext` requests;
- `getbulk-request` -Implemented as a sequence of `GetNext` requests, retrieving large blocks of data, such as multiple rows in a table;
- `get-response` -used by the device agents as replies to `Get`, `GetNext` and `Set`. It acts as a delivery mechanism for the requested information;
- `set-request` -changes configuration variables in the agent device;
- `trap` -unsolicited messages sent out by SNMP agents alerting the SNMP manager to a condition or event. These `trap` messages concern the

failure of the monitored device, maintenance issues, and other unexpected conditions;

- `inform` -the SNMP manager's version of a `response` and is a reply to a `trap` message.

The SNMP manager solicits `get-request`, `set-request`, `getnext-request`, and `getbulk-request` PDUs, while the SNMP agent responds with `response` PDUs. The SNMP agent responds to the SNMP manager requests by means of the `get` and `set` PDUs. With the `get-request`, the agent retrieves the value of the requested MIB variable and responds to the manager with that value. While the `set-request` forces the SNMP agent to change the value of the MIB variable to the value specified by the manager. The following figure illustrates a SNMP `get-request` to determine the operating status of the router G0/0 interface.

Figure 2.3 - Example of a SNMP GetRequest.

The Trap command is a special feature of the SNMP agent because it will notice events that the scheduled Get requests from the manager might miss in the polling interval. It is possible for SNMP agents to generate and send traps to inform the manager immediately of these certain events. Trap-directed notifications reduce network and agent resources, by eliminating the need for some of SNMP polling requests. Figure 2.4 demonstrates a SNMP trap generated by a router informing the SNMP manager that the G0/0 interface is down.

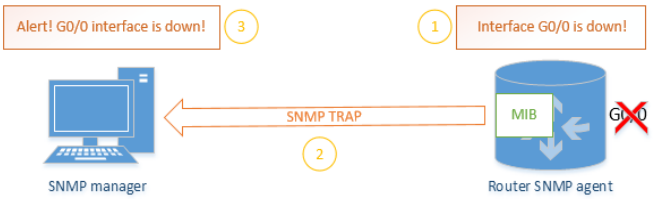


Figure 2.4 – Example of a SNMP trap.

There are several versions of SNMP, which can be a problem when buying network equipment or choosing one to implement. The existing versions are:

- SNMPv1 - The Simple Network Management Protocol, a Full Internet Standard, defined in RFC 1157 [2];
- SNMPv2c - Defined in RFCs 1901 [3] to 1908 [4]; utilizes community-string-based Administrative Framework;
- SNMPv3 - Interoperable standards-based protocol originally defined in RFCs 2273 [5] to 2275 [6]; provides secure access to devices by authenticating and encrypting packets over the network.

Both SNMPv1 and SNMPv2c use a community-based form of security, which is a minimal security measure involving a *cleartext* string known to a group of entities. SNMPv1 is a legacy solution, not encountered in modern networks and substituted be SNMPv2c. SNMPv2 has a bulk retrieval mechanism and more detailed error message reporting to the management stations.

SNMPv3 is the most secure option because of its encrypted messaging and more sophisticated authentication procedures. The message integrity feature ensures that a packet was not altered in the transmission; authentication determines that the message is from a valid source, and encryption prevents the contents of a message from being read by an unauthorized source. Table 2.1 compares the various existing SNMP security models and levels.

Table 2.1 - SNMP version comparison.

Version	Level	Authentication	Encryption	Observations
SNMPv1	noAuthNoPriv	Community string	No	community string based authentication
SNMPv2c	noAuthNoPriv	Community string	No	community string based authentication
SNMPv3	noAuthNoPriv	Username	No	username based authentication
SNMPv3	AuthNoPriv	MD5 / SHA	No	Secure authentication, message integrity
SNMPv3	AuthPriv	MD5 / SHA	DES / AES	Secure authentication, message integrity, message encryption

Network devices already have SNMP capabilities installed, but in some cases, the service may need to be activated. After configured correctly, the SNMP service daemon will start up with the device operating system.

The SNMP manager consists of three elements: collector, a data store and a user interface. Typically, the data store is an SQL database, the user interface is a web GUI and the collector is the service in charge of querying the SNMP agents. All these elements can reside on the same server for a simplified installation but can also be divided into different servers for more powerful processing. When the SNMP agents on the relevant network devices are configured and running, they are ready to respond to the Get requests sent by the manager collector service. The collector service will then gather all the retrieved monitoring data and store it in the SQL database. The network administrator can then visualize this data in the interface GUI.

2.3. Network monitoring tools

Managing a network using the right tools may be the difference between having an efficient network or a network that merely responds to the user's load. While network downtime, outages and slow performance might lead to losing vital business transactions, efficient proactive monitoring can help predict and prevent network outage. Due to this reason, network monitoring tools should be an integral part of every IT admin's arsenal. Therefore, choosing the right monitoring tool to implement in a network is a crucial and

daunting task, and it is useful that this choice is done only once or as fewer times as possible.

There are several SNMP monitoring systems available on the market today, ranging from proprietary paid platforms all the way to open source freeware options. Network monitoring systems can cover a wide array of functionalities. Some software focuses more on observation and monitoring of network traffic and others are more oriented on response and awareness, with features for setting alarms and triggers for specific events. There are even systems that generate traffic trends and network performance statistics for long-term adjustments to the hardware devices.

So choosing the right monitoring system for the university's network topology, keeping in mind the strict budgeting limitation, was of the utmost importance. The first step would be to think of all the features that would be important to have in a monitoring system, based on the various years of experience administrating the university's network. The solution would have to be compatible with the university's multivendor network and be scalable enough to permit future growth. It should also have a good alerting system to provide an effective approach to solving issues. Some requirements were mandatory, *e.g.*, no purchase cost nor licensing fees, SNMP support, and the presence of an active technical community in the web. Other requirements were optional, but nice to have, such as good and intuitive web interface and flexible reporting tools. Therefore, the following criteria would have to be met in the monitoring solution:

- No cost solution
- Open Source supported by a community
- Powerful and scalable enterprise solution
- Good support and frequent releases
- SNMP based monitoring
- Good web GUI
- Multi-platform SNMP agents

- Good documentation
- Template configuration
- Alerts and notifications, via email and SMS
- Good Report and Graph tools
- Possibility to monitor over 500 devices
- History and trend features
- Support for Auto Discovery features.

With all these considerations in mind and after some online research for the best open source monitoring tools, it was noted that the same software platforms kept appearing in several articles featured in reputed web sites well known among network administrators, *e.g.* on ittsystems.com [7], opensource.com [8], techwiser.com [9], blog.capterra.com [10] and pcwdld.com [11]. A pre-selection was made by removing the ones that did not meet the above criteria (*e.g.* not open source, no SNMP agent, bad support, fragility when updating) or the ones oriented for smaller or home use implementations (*e.g.* free version limited to 100 devices). Some tools were also improved versions (forks) of existing older ones, so it was also decided to analyze the newer ones instead. Another limiting factor was the difference in the number and type of features between the open source version and the enterprise paid version that some companies offered, giving the impression that the free versions were not robust enough for large organizations. The following list summarizes the best contenders:

- **Zabbix** (https://www.zabbix.com) described as "(…) is the ultimate enterprise-level software designed for real-time monitoring of millions of metrics collected from tens of thousands of servers, virtual machines and network devices. (…)" [12]. Zabbix is open source and with no licensing cost and offers real-time monitoring of up to 10000 hosts. It is easy to install and configure via the web interface, with compatibility across numerous systems and environments. It also comes with a modern and intuitive interface. Other features include: monitoring templates, API, agents for all platforms, alerting,

reporting, interesting graph generating feature and network mapping abilities [12];

- **Nagios Core** (https://www.nagios.org/projects/nagios-core/) is the free open source version of Nagios XI. Recognized as one of the most popular and older tools in the market, some users mentioning that it is "(…) extremely fast and incredibly reliable (…)" [13], though the user interface of Nagios looks somewhat outdated. By default, all the configuration (*e.g.* adding hosts and services to be monitored) for Nagios is done through text files. This can take some time to get used to, resulting in a steep learning curve, especially for beginners. Out of the box, Nagios is not an evident choice due to the text-based configuration and an outdated web interface. However, the numerous plugins and scripts available for Nagios makes it highly customizable. Users report than "(…) Nagios can track, monitor, and follow trends of almost any data that can be gathered from a network environment, and it is that same flexibility and wide-open capabilities that are both its strength and its main drawback (…)" [11].

- **Icinga** (https://icinga.com/) is a fork of the Nagios monitoring tool, but with a completely rewritten core to increase its responsiveness, reduce complicated setups, and allow easy use. Out of the box, Icinga looks much better with its responsive web user interface, also having extensive database support and much better scalability then Nagios. Features include monitoring of both live and historical performance data, event handlers, customizable template-based reports, plugin support and graphs for analysis and data manipulation [14]. Icinga still uses text files for configurations, but the process is improved when compared to Nagios.

- **Libre NMS** (https://www.librenms.org) is a fork of another monitoring tool called Observium. LibreNMS is a complete free open-source

network monitoring tool based on SNMP, supporting a broad range of operating systems and network devices. It provides features such as interesting graphs, configuration via web interface, auto network discovery, alerting (SMS, e-mail), API and plugins [15].

The first step was to visit each of these products' web site and search for product documentation, screen shots, demos, forum support and overall technology features. The second step was to organize these platforms in a product comparison chart in order to compare the different features and to help analyze the pros and cons between them. An assessment of each feature was made and a value from 1 to 5 was attributed to each product feature, being 1 the worst and 5 the best. Table 2.2 shows the results. The table lines contain the assessment for the monitoring software and the columns indicate the required specifications.

Table 2.2 - Product comparison chart.

Software	Cost[a]	Configure	GUI	Graphs[b]	Reports	Templates	Agents	Scale	Com[c]	Total
Zabbix	5	4	5	4	4	5	5	4	5	41
Nagios Core	3	2	2	2	2	2	5	2	5	25
Icinga	5	3	3	3	3	2	5	3	3	30
LibreNMS	5	3	3	4	4	2	5	4	4	34

(a: purchase cost; b: Graphics and Charts; c: Technical Support Community)

It was observed that all these monitoring tools were practically equal in the features they offered. All of them had adequate graph and reporting features, a good support community and multi-platform agents. Zabbix clearly surpassed the others in terms of initial installment and configuration, user web interface and pre-installed templates. In terms of purchase cost, they are all free, unlike Nagios, which offered the better Nagios XI at a price. LibreNMS seemed to be less recognized then Zabbix in terms of search results and articles, even though it was also rich in features. Nagios being the older product, suffered from a steep learning curve, due to its text file

configuration process and outdated user web interface as in regards to Zabbix [16]. A big advantage of Nagios Core is its community and extensive plugin options. Icinga is a fork of the Nagios monitoring tool and has a better responsive web user interface, extensive database support and better scalability than Nagios. An interesting article found online describing the reasons why one company chose to switch from Icinga to Zabbix was also taken in account [17]. The article stated that the reasons for switching was the ease of use and richness in features.

Besides the total points achieved, a deciding factor for choosing Zabbix over the other platforms was the positive feedback received from other network administrators who were using this software in their institutions. Factors such as stability, fast response, frequent updates and good graphics were the most praised. The fact that Zabbix offers only one enterprise solution, not differentiating between free and paid versions (Zabbix only charges for technical support) was also taken in account.

2.4. Zabbix

Zabbix is a distributed monitoring tool with a central web interface and is composed of three components: the Zabbix server, the database and the web interface. These three services can reside on the same server, but for large environments, it is better to use a dedicated server for each one of these components to allow less server load and better service isolation.

Zabbix can be setup with three possible architectures as shown in figures 2.5, 2.6 and 2.7.

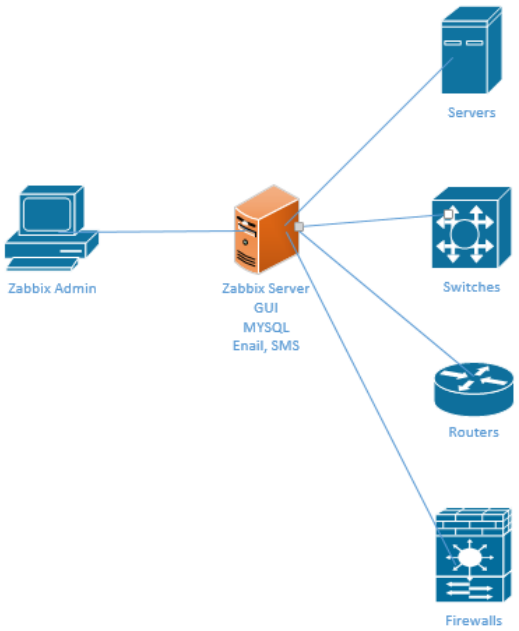


Figure 2.5 – Zabbix simple architecture.

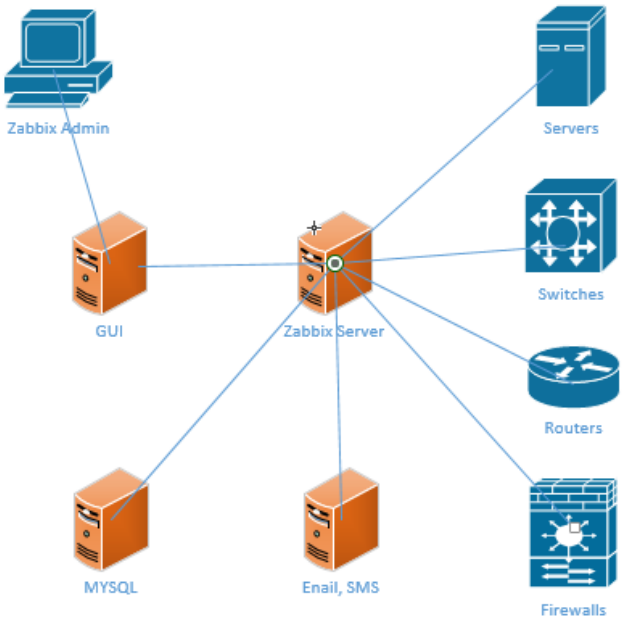


Figure 2.6 – Zabbix complete architecture without a proxy.

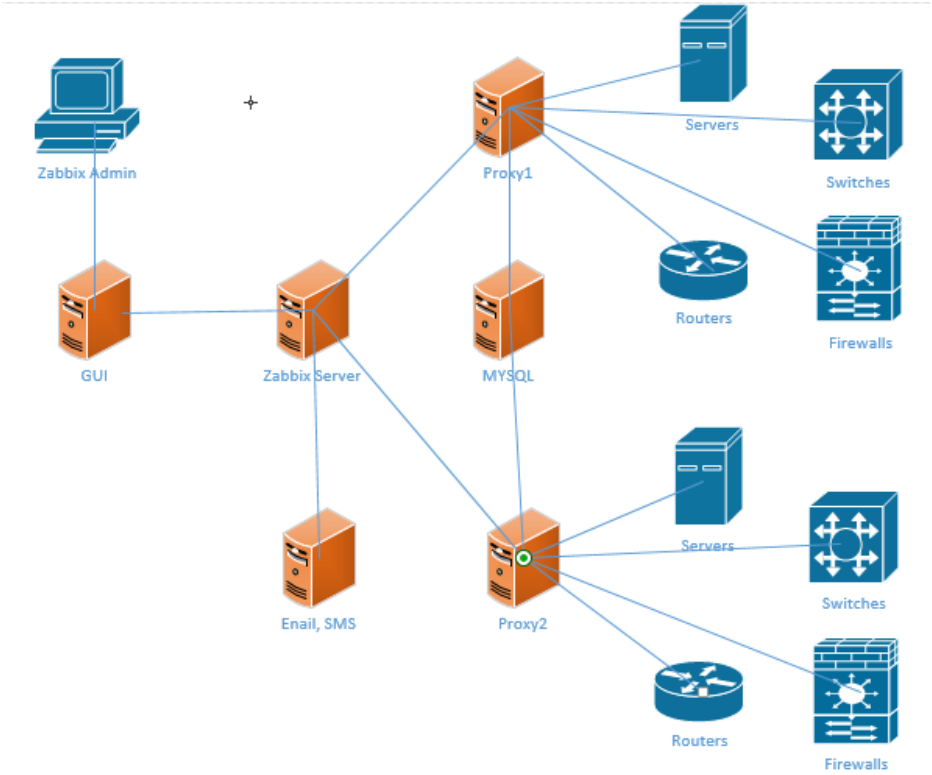


Figure 2.7 – Zabbix complete architecture with a proxy.

Zabbix uses a client-server architecture and gathers data from the SNMP agents running in the network devices. Zabbix can take input data also from a Zabbix proxy. The proxies can collect data on behalf of the Zabbix server, save it in a local buffer and then send it to Zabbix server. The advantage is that the proxy will remove some of the processing load from the Zabbix servers, particularly for large environments where many hosts are being monitored and different proxies can be setup to partition the SNMP data.

Zabbix can also be configured to receive SNMP traps, which are the opposite to querying SNMP-enabled devices. The trap is sent from a SNMP enabled device and is collected or “trapped” by Zabbix. Traps are sent upon some condition change in the device, so the advantage is detecting issues that the scheduled Get requests from the manager might miss in the polling interval.

Some of the following Zabbix terminologies will help to understand some of the concepts described in the following chapter dedicated to the implementation of Zabbix:

- Action – reaction to an event;
- Discovery – the scanning of IP ranges to discover hosts;
- Event – occurrence of something that deserves attention;
- Frontend – Zabbix web user interface;
- Graphs - visual representation of the host data;
- Host –device that should be monitored;
- Host group – hosts are organized into host groups;
- Item – a piece of data received from a host;
- Network map – visual representation of the network infrastructure;
- Screens – grouping of information from various sources to a single screen or table;
- Template – a set of entities (items, triggers, applications, low-level. discovery rules, graphs, screens, web scenarios) applied to the hosts;
- Trigger – logical expressions that "evaluate" data;
- Zabbix agent – a service running on a host to monitor locally;
- Zabbix Server – central server that collect and process the host data.

2.5. Conclusion

After having presented the basic concepts on network monitoring and the elements for the operation of SNMP, this chapter reviewed the Zabbix, Nagios Core, Icinga and LibreNMS solutions, which are free as to comply with the requirements of its intended operation scenario.

Section 2.4 concludes that Zabbix was the better tool to use in UBI's network infrastructure, as it provides all the features that are necessary, but in a better way than its contenders.

Chapter 3

3. Design and Implementation

The previous chapter concluded that the Zabbix monitoring software was the best choice for the UBI network, due to its advantages over the other software brands regarding the university's specific needs, such as free licensing, template features, graphs, reporting and frequent updates.

This chapter will focus on the steps taken in implementing the Zabbix monitoring software, as well as the decisions and troubleshooting steps encountered during the install, configuration and fine-tuning steps. The processes are described in detail, as this has proven to be a good practice among Systems and Network Administrators.

3.1. Virtualization Server

The decision that Zabbix would run in a virtualized environment was because a virtual machine would be easier to manage, maintain and backup. This would also reduce spending and be more efficient to run because the same host machine could run other virtualized environments concurrently.

Due to budgeting restrictions, the virtualization server was setup on a desktop PC with additional RAM and hard disk to support the requirements needed in terms of physical and disk memory instead of an actual dedicated server as initially thought. The PC was a HP EliteDesk 800 G1, with an Intel 4th Generation Core i7 Processor, which originally had 8GB 1600-MHz DDR3 of SDRAM and a 500GB SATA hard drive.

The Zabbix documentation in section Installation - Requirements, stated that, as minimum requirements, Zabbix would need at least 128 MB of physical memory and 256 MB of free disk space. However, the amount of required disk memory obviously depends on the number of hosts and

parameters that are to be monitored. If there are plans to keep a long history of monitored parameters, it is necessary to have at least a couple of gigabytes to provide enough free space to store the history in the database. Because each Zabbix daemon process requires several connections to the database server, the amount of memory allocated will depend on the configuration of the database engine. The documentation also stated that Zabbix and especially the Zabbix database might require significant CPU resources depending on number of monitored parameters and chosen database engine [18].

With these requirements in mind, the server was upgraded to 32GB of RAM and an additional 8TB hard drive. This increment would allow the server to host various virtual machines and provide sufficient storage to keep a long history of monitored Zabbix parameters.

After the hardware installation, it was time to install the host operating system. It was decided on the Community ENTerprise Operating System (CentOS) Linux distribution [19], because the SI department had various servers running older versions of this OS, so the know-how to install and maintain was already acquired and time has proven it to be a lightweight, fast and reliable operating system. CentOS is an enterprise grade, free open source project with the same functionality, performance and stability as the paid operating system Redhat Enterprise Linux.

In order to install the CentOS, a bootable USB drive, fully compatible with UEFI based motherboards was created. Creating a bootable CentOS USB Stick on a Windows™ machine was a relatively straightforward process using the RUFUS program [20]. The only requirements needed where:

- RUFUS freeware application for Windows™ 10
- CentOS ISO file for creating Installation Media.
- USB device with adequate capacity

The CentOS DVD ISO file was downloaded from the CentOS downloads page [21] as well as the RUFUS software. Figure 3.1 illustrates the settings used in creating the bootable CentOS USB drive:

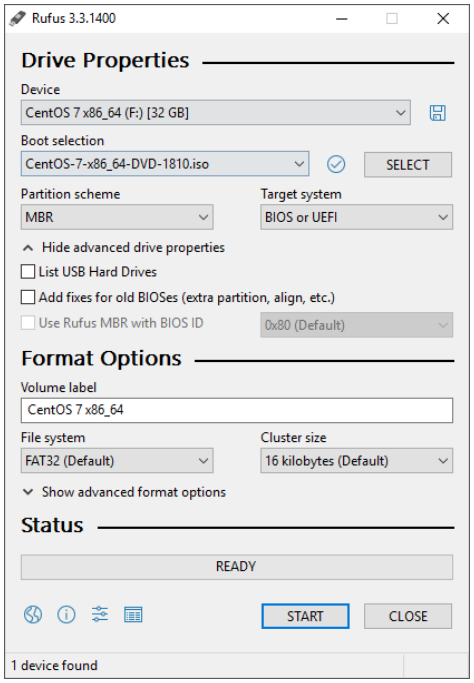


Figure 3.1 – Rufus application settings.

Before installing CentOS 7 on UEFI based motherboards, one should open the machine motherboard UEFI settings, disable the QuickBoot/FastBoot and Secure Boot options, change the machine boot order and instruct the BIOS/EFI to use the USB drive for booting.

After these steps where completed, the next step was to insert the USB drive, power-on the machine, hit the appropriate bootable key from the keyboard and instruct the BIOS or UEFI software to boot the machine from the appropriate USB drive. The CentOS 7 bootable ISO image loaded and presented the first installation image on the monitor screen with the option to install CentOS 7. After selecting this option, the installation process began. The options chosen in the installation process are briefly presented in Table 3.1 since the nature of this work is the installation and configuration of Zabbix and not CentOS itself.

Table 3.1 - CentOS7 install settings

Feature	Setting
Language	English
Localization	Portugal
Keyboard	PT
Software Selection	KDE Desktop
Installation Destination	Manual partition configuration /: 430GB /boot: 210MB swap: 68GB
Network	Manual configuration: IP: 192.168.69.5 MASK: 255.255.255.0 Gateway: 192.168.29.254 DNS: 192.168.100.1
KDUMP kernel crash	Disabled (to free system RAM)

The last step was to define the root password and create a user with administrative rights. After the installation process completed and rebooted, the system was updated with the KDE Application Installer.

After the system update, it was time to install the virtualization software. The Oracle VM VirtualBox [22] software is a good option, due to its zero purchase cost, ease of use and reliability. The installation proved to be a straightforward process using the included Centos application installer software. Finally, it was time to create a virtual machine and proceed with the installation of the Zabbix software, but this process will be described in the following section.

3.2. Zabbix virtual machine

Once the CentOS 7 host machine was installed, setup with the latest updates and running Oracle VirtualBox, it was necessary to create a virtual machine for the installation of Zabbix. The operating system on which Zabbix would run would once again be CentOS 7, but this time with the minimal ISO image

[23]. A minimal install option is the most suitable environment for a server since it is the most flexible option with the smallest disk footprint. There was no need for graphical interfaces and desktop applications since the only software needed was Zabbix, MySQL and Apache. Any additional package or software could be installed as needed.

The creation of the virtual machine with Oracle VirtualBox is a simple process, but will not be described on account of it being beyond the scope of this work. The settings chosen for memory, CPU and hard drive would have to be in accordance to the number of monitored hosts of the UBI network infrastructure. The Zabbix documentation stated the following hardware configuration examples shown in Table 3.2.

Table 3.2 - Examples of hardware configuration.

Network	Platform	CPU/Memory	Database	Monitored hosts
Small	CentOS	Virtual Appliance	MySQL InnoDB	100
Medium	CentOS	2 CPU cores/2GB	MySQL InnoDB	500
Large	RedHat Enterprise Linux	4 CPU cores/8GB	RAID10 MySQL InnoDB or PostgreSQL	>1000
Very large	RedHat Enterprise Linux	8 CPU cores/16GB	Fast RAID10 MySQL InnoDB or PostgreSQL	>10000

The UBI network infrastructure fitted in the Medium option, but to anticipate increments in the number of monitored hosts and minimize performance issues, the virtual machine was configured with 4 CPUs, 8GB memory and 2TB hard disk. The image was created in the added 8TB hard drive to ensure that enough storage was available for the expected growth. Figures 3.2, 3.3 and 3.4 illustrate the settings used in creating CentOS 7 virtual machine.

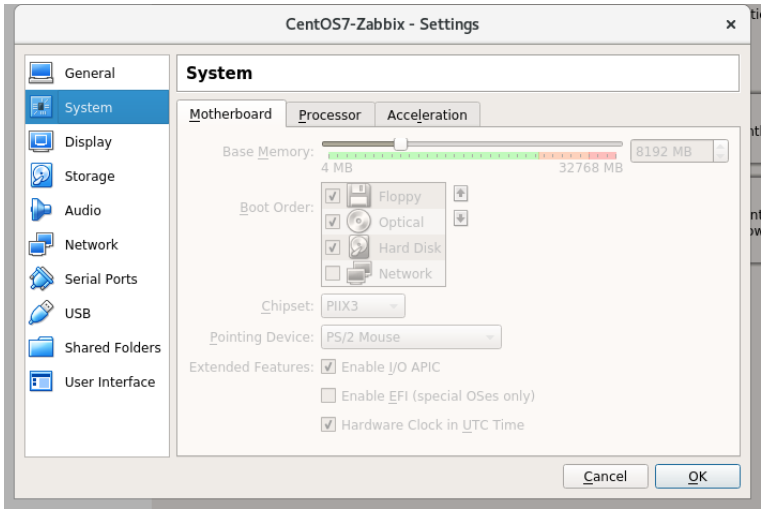


Figure 3.2 – VirtualBox memory settings.

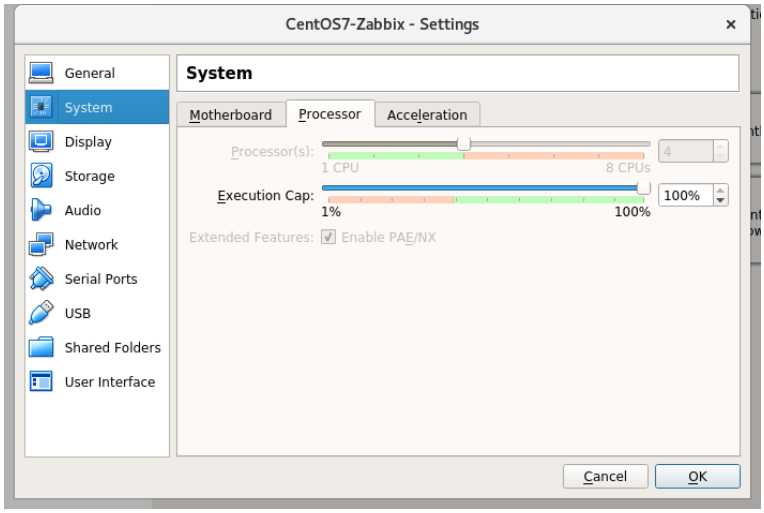


Figure 3.3 – VirtualBox CPU settings.

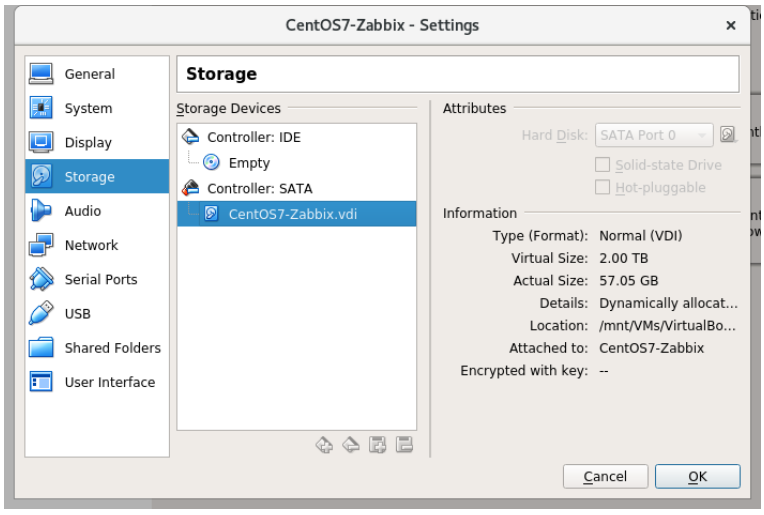


Figure 3.4 – VirtualBox storage settings.

Once the virtual machine was running, it was time to update the system and install additional packages for managing the operating system. Since the OS installed was the minimal install version, with no graphical interface, the update and installations would be accomplished through CLI. The `yum` command is the primary tool for getting, installing, deleting, querying, and managing Red Hat Enterprise Linux RPM software. Table 3.3 lists the applications installed with the `yum install` command.

Table 3.3 - Installed applications

Application	Description
Joe	powerful ASCII-text screen editor
epel-release	repository that provides easy access to install packages for commonly used software
wget	command-line utility which downloads files over a network
net-tools	networking tools

3.3. Zabbix installation

This section covers the steps in installing and configuring the open source monitoring system Zabbix 4.2 on CentOS 7. Various install procedures guides were found on the Internet, but the most helpful websites were the following: Zabbix documentation [24], Computing for Geeks [25], Omgfoss [26] and How to Forge [27]. The following procedure where based on these resources.

Zabbix depends on the following software applications:

- Apache web server
- PHP with required extensions
- MySQL/MariaDB database server

The web server packages, as well as the scripting language PHP and MySQL database server packages can be installed from the default CentOS 7 repository, utilizing the `yum install` command.

It was recommended in one of the install guides to have SELinux in permissive mode to avoid any issues that could arise during the installation and configuration process. SELinux is a security feature of the Linux kernel, which has proven to be difficult to understand and configure without the required skills. Therefore, SELinux was set in permissive mode, which still provided useful security logging.

The install procedures and operating system configurations are outlined in the following six steps.

3.3.1 Install and Configure Apache `httpd` server

The first step was to install the Apache web server with the following *superuser* privileged command:

```
# sudo yum –y install httpd
```

After installing Apache, basic security settings where configured by adding a few lines ate the end of the Apache configuration file. The `joe` text editor program was used for editing files.

```
# joe /etc/httpd/conf/httpd.conf
ServerSignature Off
ServerTokens Prod
ServerName zabbix.ubi.pt
```

The `ServerSignature` directive instructs Apache not to display the server version, operating system details, installed apache modules, PHP-version, etc on error pages. The `ServerTokens` directive configures the HTTP response header to only return the text *Apache*, hiding other sensitive data.

Now it was necessary to restart Apache and allow the communications through the firewall.

```
# sudo systemctl restart httpd
# firewall-cmd --zone=public --add-service=http –permanent
# firewall-cmd --reload
```

3.3.2 Install and Configure PHP

After the Apache install, it was necessary to install PHP, edit a few settings and restart the Apache `httpd` service.

```
# sudo yum -y install php php-pear php-cgi php-common php-mbstring
php-snmp php-gd php-xml php-mysql php-gettext php-bcmath

# joe /etc/php.ini
 date.timezone = Europe/Lisbon

# sudo systemctl restart httpd
```

3.3.3 Install MariaDB Database server

Now it was time to install the database server, start the service and allow its connections through the firewall.

```
# yum install mariadb-server
# systemctl start mariadb
# systemctl enable mariadb
# firewall-cmd --zone=public --add-port=3306/tcp --permanent
# firewall-cmd --reload
```

The `mysql_secure_installation` script was run in order to configure the SQL server security settings, such as creating the root password, dropping the test database and access privileges.

```
# mysql_secure_installation
```

Once the database server was installed and configured, it was time to create the Zabbix database and grant user permissions. For security purposes, the user password has been substituted with x's as shown in the following screen shot.

```
# mysql -uroot -p
MariaDB [(none)]> create database zabbix character set utf8
collate utf8_bin;
MariaDB [(none)]> grant all privileges on zabbix.* to
zabbix@localhost identified by 'xxxxxx';
MariaDB [(none)]> FLUSH PRIVILEGES;
MariaDB [(none)]> quit;
```

3.3.4 Install Zabbix 4.0 Server

Now that the required dependencies where installed and working, it was time to finalize the installation by deploying Zabbix server from the Zabbix repository.

```
# yum -y install
http://repo.zabbix.com/zabbix/4.2/rhel/7/x86_64/zabbix-release-
4.2-1.el7.noarch.rpm
# yum install -y zabbix-server-mysql zabbix-web-mysql zabbix-agent
zabbix-ge
```

After the installation, it was necessary to import the initial schema and data for the server with MYSQL:

```
# zcat /usr/share/doc/zabbix-server-mysql*/create.sql.gz | mysql -
uzabbix -p xxxxxx
```

3.3.5 Configure and start Zabbix server

After the database setup, it was necessary to edit the Zabbix configuration file `/etc/zabbix/zabbix_server.conf` and set database connection settings:

```
# joe /etc/zabbix/zabbix_server.conf
DBName=Zabbix
DBUser=Zabbix
DBPassword=xxxxxx
```

Some more configurations for the Zabbix frontend were also necessary. The Apache configuration file for the Zabbix frontend is located in `/etc/httpd/conf.d/zabbix.conf`. Some PHP settings are already configured, but it was necessary to uncomment the `date.timezone` setting and set the right time zone for Portugal. The recommended settings were:

```
# joe /etc/httpd/conf.d/zabbix.conf
php_value date.timezone Europe/Lisbon
php_value max_execution_time 300
php_value memory_limit 128M
php_value post_max_sizoe 16M
php_value upload_max_filesize 2M
php_value max_input_time 300
php_value max_input_vars 10000
```

After changing these values, it was necessary to open some TCP ports in the firewall for the Zabbix agent and trapper process. After opening the port on the firewall, it was necessary to restart the `httpd` and `zabbix-server` processes.

```
# sudo firewall-cmd --add-port={10051/tcp,10050/tcp} --permanent
# sudo firewall-cmd --reload
# sudo systemctl restart httpd zabbix-server
# sudo systemctl enable zabbix-server
```

3.3.6 Zabbix initial setup

The last step was to perform the initial Zabbix setup. This is accomplished through the Zabbix frontend, available at `http://zabbix.ubi.pt/zabbix/`. This process was completed by following six steps:

1. In the welcome page, click `Next Step` button to proceed;
2. A confirmation page that all pre-requisites were satisfied;
3. Database configuration page to insert the DB connection parameters:
    - database type: MYSQL
    - Host: localhost
    - Port: 0 (use default port)
    - Name: zabbix
    - User: zabbix
    - Password: xxxxxx
4. Zabbix server details page:
    - Host name: localhost
    - Port: 10051
    - Name of the installation: zabbix.ubi.pt
5. Verify all settings page and click Next step to finish the initial setup, which presented a congratulations page and click to `Finish` button to end the installation. This action led to the login page for which the default credentials were: Username: admin and Password: zabbix;
6. After the successful login, it was now necessary to change the default password for security reasons. This was accomplished navigating to `Administration>Users>Admin>Password>Change Password`.

3.4. Zabbix configuration

Once the Zabbix server was installed and ready for monitoring, some preparation work was needed before adding hosts. This would be the ideal time to standardize the UBI network infrastructure in regard to device host names and SNMP information, such as location, contacts, system names *etc*. It was necessary to establish a naming convention for monitored hosts, which contained valuable information such as IP address, friendly name and device brand.

An important issue was addressed as to what SNMP version would be implemented on the monitoring system. The choice was between SNMPv2c and SNMPv3. Best practices indicate that SNMPv3 would be the correct choice due to its security mechanisms. Almost all of the university's network devices support both versions, except the older Enterasys VH models, which only support SNMPv1. However, the decision was to use SNMPv2c for two reasons. The first one being that the management IP addresses of every network device is on a secure VLAN that only accepts communications from the network administrator's subnet where Zabbix is also running. This guarantees that nobody else can listen to network management traffic, including the SNMP protocol messages. The other reason, and perhaps the most relevant, was that it is easier to troubleshoot SNMP issues while listening to unencrypted messages. Of course, SNMPv3 PDUs do not need to be encrypted, but still, the additional overhead of the more secure protocol was not needed in the current setup. Zabbix templates can be configured to run any SNMP version, so in the future if a device outside of the secure VLAN was added, then a SNMPv3 template could be configured to address the issue. The older VH models would need a SNMPv1 template

After establishing the naming convention and changing all of the network device's SNMP variables, the next step would be to start configuring and populating Zabbix with host devices. Following the Zabbix documentation, it

was concluded that the following steps would have to be taken in order to successfully add hosts:

- Configure User Groups and Users
- Configure Media Types (emails)
- Configure SNMP Community
- Configure Host Groups
- Configure Templates
- Configure Discovery and Actions.

These steps will be explained in further detail (with accompanying screen shots) throughout the course of this chapter.

3.4.1 Naming conventions

As explained in the previous chapter, SNMP is a network protocol created to provide a consistent and reliable way for different devices on a network to share information with one another. It allows network management tools to identify devices, monitor network performance, keep track of changes to the network, or determine the status of network devices in real time. In order to correctly and unequivocally identify these network devices in the Zabbix platform, it was necessary to analyze the current naming conventions used in the UBI network infrastructure and if necessary, find a way to improve and standardize this information

The UBI network infrastructure is currently managed with the Enterasys NetSight Console software, which provides a wide selection of advanced networking management and analytic tools. Netsight Console allows the management of the entire network through a single interface and monitors the status of network devices such as switches and routers with the SNMP protocol. To allow future comparison with the new Zabbix interface, Figure 3.5 shows the main Netsight interface.

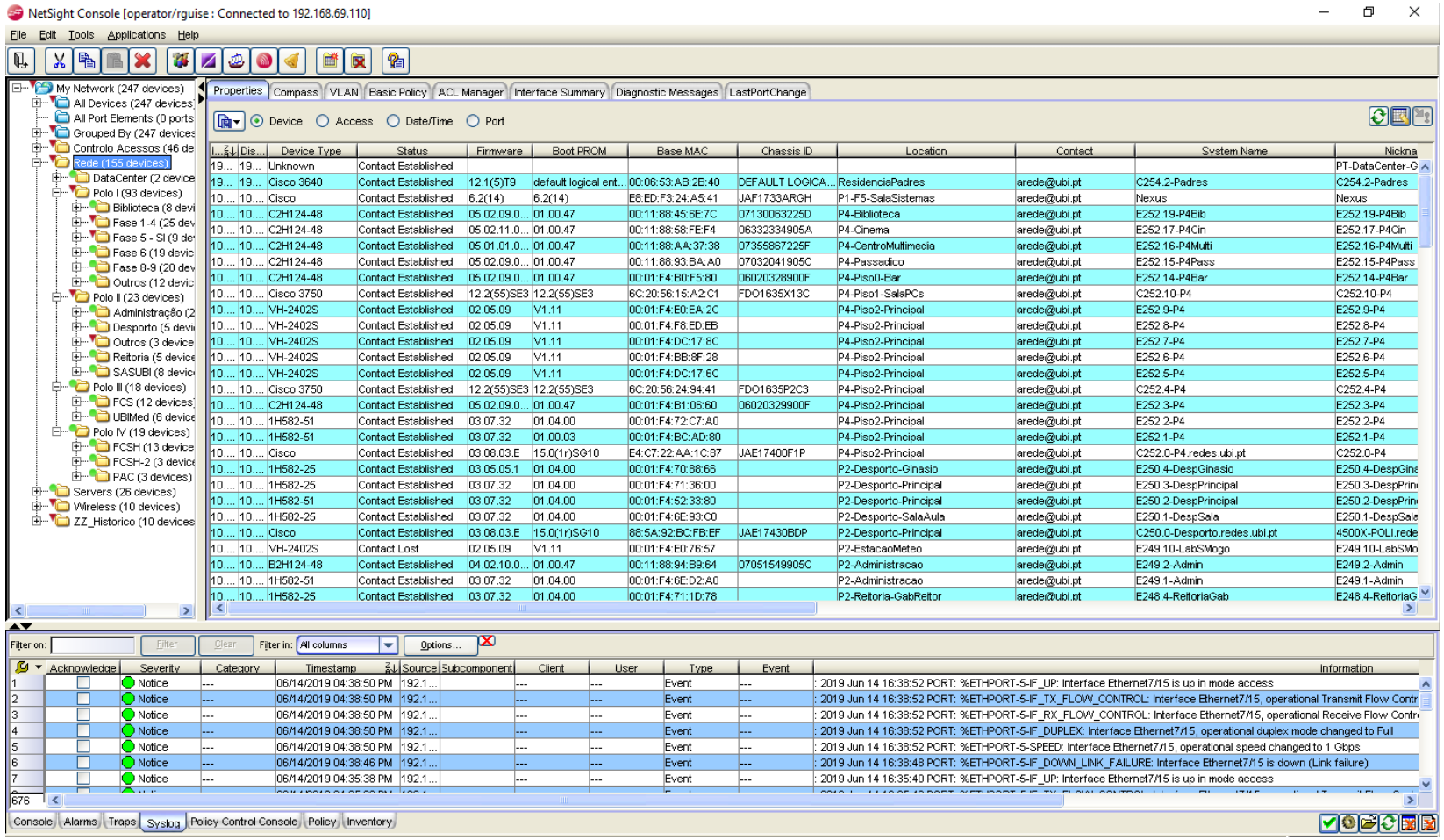


Figure 3.5 - Netsight Console main window.

For organizational purposes, the list of monitored devices available on the left side of the window was divided into hierarchical geographical zones that start with the UBI campuses and ends with the building number. Presently there are 154 networking devices being monitored as partially shown in Figure 3.6.

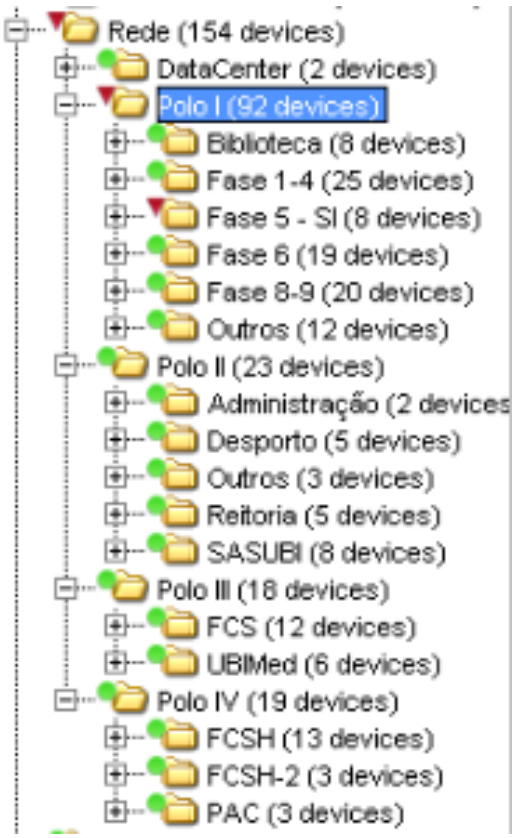


Figure 3.6 – NetSight monitored devices grouped by location.

Figure 3.7 highlights the SNMP contact, information and system name that are configured in some of the network devices.

Location	Contact	System Name
SI-SalaRedes	SIUBI	2960x-SalaRedes.redes.ubi.pt
Textil		Textil1
		Textil2
CFIUTE	CIUBI	
Malufa	CIUBI	MALU-EME1-01
academicos		
ServicosAcademicos	Redes@SIUBI	2960x-Sacad.redes.ubi.pt
SalaInqueritos	CIUBI	
ATEL	CIUBI	ATEL1-EME1-01
CREA1	CIUBI	CREA1-EME1-02
CREA1	CIUBI	
CREA	CIUBI	CREA-E1
Parada Sala Profs		
ATEL	SIUBI	ATEL1-EME1-02
Servicos Graficos	CIUBI	CIUBI
PAPEL	CIUBI	PAPEL-EME1-01
Departamento de Letras	CIUBI	
SALA TERMINAIS I	CIUBI	FAS4-EME1-01
SALA TERMINAIS II	CIUBI	FAS4-EME1-02
bastidor 2 piso IV fase	CIUBI	FAS4-EME1-03
Bastidor piso 2 IV fase (ampliação)	CIUBI	E1-DFISICA-1
SALA TERMINAIS II	CIUBI	
Centro de optica	CIUBI	Centro de optica
Polo1-FisicaP5	CIUBI	TPL-FisicaP5
Fisica-Secretariado	SIUBI	2960x-FisSec.redes.ubi.pt
CORREDOR PISO 3	CIUBI	VH-DFISICA-2
Fisica-4piso	SIUBI	2960x-Fisica4P.redes.ubi.pt
Sala Terminais	SIUBI	2960X-Quimica.redes.ubi.pt
Sala de Sistemas	CIUBI	EME7-CINFed1-01
SI-SalaSistemas	SIUBI	2960x-SIUBI-1.redes.ubi.pt
SI-SalaSistemas	SIUBI	2960x-SIUBI-2.redes.ubi.pt
SalaSistemasEVA		
HALL-SALA-DE-REDES	CIUBI	CI-VH-HALL-SALA-DE-REDES
SI-SalaSistemas	SIUBI	VPN_AMA.redes.ubi.pt
Fase 6 Piso 2	CIUBI	Fase6-1
	CIUBI	Fase6-2
	CIUBI	Fase6-3
		Fase6-4
UBI-Polo 1-Fase 6	CIUBI	Fase6-1
Fase6-P2	SIUBI	Fase6-S6.20

Figure 3.7 – Netsight device SNMP values.

Here it can be seen that a naming convention was not followed in regard to system name and locations. Some of the devices did not have system names defined, others had different contact information and the locations were not standardized. Clearly, it was necessary to standardize all this information before starting to populate the Zabbix software.

The first step was to identify all the locations that contained network device racks, which would be organized by the campus geographical location; building or construction phase; building floor or room or service/department if applicable. This information, configuring a hierarchical coding scheme, would be used to populate the device's SNMP location information. The conclusions of this study is presented in the following table:

Table 3.4 – Network Device Locations.

Campus	Building/Phase	Service/Floor/Room
Polo1	Fase1	Academicos, Sala Inqueritos, Letras
Polo1	Fase2	Labcom, Crea, Crea Gab, Sala Prof, Graficos
Polo1	Fase3	Textil, Papel
Polo1	Fase4	Terminais, Terminais AT, Física Sec, Fisica Ampliação, CO, Fisica P3 Corredor, Fisica P4, Fisica P5
Polo1	Fase5	SI, Hall
Polo1	Fase6	Piso 1, Piso 2, Sala Sistemas, IT, Release
Polo1	Bloco8	Sotao, Escadaria
Polo1	Bloco9	F.Gomes, Aero, Luciano, Sala9.2, Principal
Polo1	Bloco10	Arquitectura, AFTEBI, Lab
Polo1	CFIUTE	CFIUTE
Polo1	Outros	Biblioteca Central P1, Biblioteca Central P3, Museu Veiga, Farmaceuticas, Malufa, ServTecnicos
Polo2	SASUBI	Sede, Armazem, MeloCastro, ResFem, ResLaranja, SantoAnt, Res6
Polo2	Reitoria	Gab Reitor, Principal, ResDoc
Polo2	Desporto	Principal, Sala, Ginasio
Polo2	Administracao	Principal
Polo2	Outros	Lab,Padres
Polo3	FCS	0AB, 0C, 0DE, 1AB, 1BC, 1DE, Core, -1AMicro, -1AServ, Portaria
Polo3	UBIMedical	P0 Principal, P0 Empresas, P1
Polo4	FCSH	Principal, Piso0, Passadiço, SalaPCs
Polo4	FCSH2	Multimedia, Cinema, Biblioteca

Following this structure, an example of the location SNMP variable of a networking switch would be *P1-F6-Piso2-SalaSistemas* where *P1* identifies the Polo 1 campus, F6 identifies the building phase, *Piso2* identifies the

building floor. The text *SalaSistemas* is the room where the network rack is located.

Another important SNMP definition is the system name. This description should provide an easy and identifiable name to facilitate the visualization of the device information when added to the Zabbix system. The description will also appear in the system terminal interface prompt, so it should unequivocally identify the device. This will reassure the system administrator that the device to which he/she is connected is the correct one.

After some thought as to what information the system name should contain, while not forgetting to maintain it at a short length, it was concluded that the description should contain the following information:

- a letter identifying the network device manufacturer
- the last two octets of the management IP address
- friendly name that was allusive to its location

An example of a system name and cli prompt is *E229.26-F6-Sistemas*, where E identifies the system manufacturer Enterasys, 229.26 are the last two octets of the IP address and F6-Sistemas is a friendly and identifiable name.

The contact SNMP variable was also necessary to define. This value identifies the person who manages the device so it would be the SI department networking area's email: *arede@ubi.pt*.

Only one more setting was missing, the SNMPv2 community string, which is a sort of user id or password that allows access to the device's statistics and this would be the defined by the value *Redes@SIUBI*.

Once the naming convention was defined, it was necessary to start changing the SNMP information of UBI's 156 networking devices. This was a time consuming process due to the nature of UBI's multi-vendor environment.

Different CLI commands were used depending on the device brand. It was also necessary to clear the previous SNMP configurations.

The following are examples of the SNMP commands used in different CLI environments to configure the devices with the new information. The manufacturer and model name are identified in bold.

Enterasys E1 model:

```
set community Redes@SIUBI ro
set prompt E229.26-F6-Sistemas
set system name E229.26-F6-Sistemas
set system location P1-F6-SalaSistemas
set system contact arede@ubi.pt
```

Enterasys B and C models:

```
set snmp community Redes@SIUBI
set snmp access Redes@SIUBI security-model v2c read All notify All
nonvolatile
set snmp group Redes@SIUBI user Redes@SIUBI security-model v2c
nonvolatile
```

Enterasys E7 model:

```
set snmp community Redes@SIUBI
set snmp access Redes@SIUBI security-model v2c read All notify All
nonvolatile
set snmp group Redes@SIUBI user Redes@SIUBI security-model v2c
nonvolatile
```

HP:

```
snmp-server community Redes@SIUBI restricted
snmp-server contact arede@ubi.pt
snmp-server location P3-1A-Micro
hostname P3-1A-Micro
```

Alcatel:

```
session prompt default A0.31-Principal#
system contact arede@ubi.pt
system location P3-UBIMedical-Piso0-Principal
system name A0.31-Principal
aaa authentication snmp local
snmp security authentication set
snmp community map Redes@SIUBI user snmpv2user on
snmp community map mode on
user snmpv2user password xxxxxx read-only all no auth
```

TPLink:

```
hostname TP227.9-FisicaP5
location P1-F4-Piso5
contact-info arede@ubi.pt
snmp-server community Redes@SIUBI read-only viewDefault
```

Cisco IOS:

```
ip access-list standard snmp_permit
permit 192.168.69.0 0.0.0.255
permit 193.136.67.64 0.0.0.15
snmp-server community Redes@SIUBI RO snmp_permit

hostname C1.1-Redes
snmp-server location P1-F5-Datacenter
snmp-server contact arede@ubi.pt
```

Cisco NX:

```
ip access-list snmp_permit
  10 permit udp 193.136.67.64/28 any eq snmp
  20 permit udp 192.168.69.0/24 any eq snmp
  30 deny ip any any
snmp-server community Redes-SIUBI use-acl snmp_permit
```

Some issues were encountered during this SNMP configuration phase. The first one being that SNMP version 2 on older devices (*e.g.* Enterasys VH

models) was not supported. The second issue was that the @ symbol in the chosen community string *Redes@SIUBI* was not allowed on the Cisco NX model. The only alternative was to continue with SNMPv1 on the older models and change the offensive symbol to a dash (-) just for the NX operating system. These issues would have to be addressed when defining the Zabbix templates and macros in the following configuration phase.

Zabbix supports the use of macros or variables that are defined globally and save time when configuring new hosts. Macros are used in templates, item key parameters and are user definable. In order to be able to use the default community string *Redes@SIUBI* across Zabbix, it would be necessary to create a user macro. This is done by accessing the menu `Administration>General`, choosing `Macros` from the drop down list and `Add` a new macro value as shown in Figure 3.8.

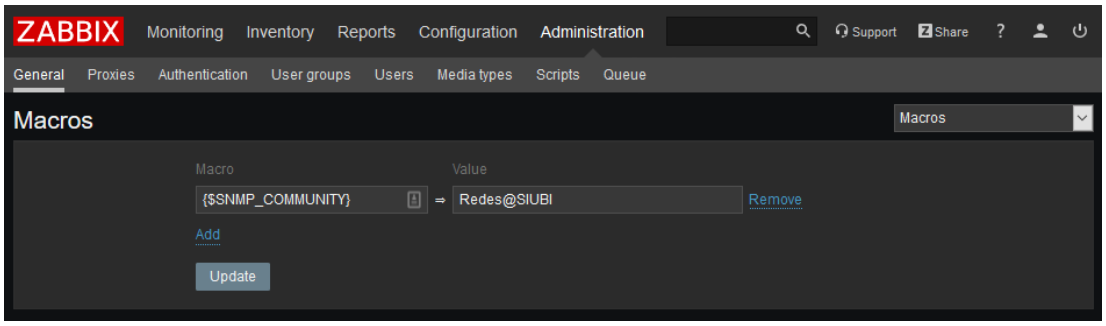


Figure 3.8 – Zabbix macros configuration page.

From now on, whenever a new host was added, this macro value would be automatically inherited, thus permitting Zabbix to access the SNMPv2 host agent.

3.4.2 Host groups

Typical Zabbix hosts are the devices the network administrator wishes to monitor (servers, workstations, switches, *etc.*), so creating hosts is one of the first monitoring tasks in Zabbix. Hosts are organized into host groups and a host must belong to at least one host group [28]. It was decided that the network devices would be grouped by equipment model and operating

system, namely due to the template specifications required for each device type. In this way, the same template would be applied to each host group, thus simplifying the host creation.

The Enterasys Netsight software allows the user to group the monitored hosts by device type, rapidly identifying the number of host groups to create as shown Figure 3.9.

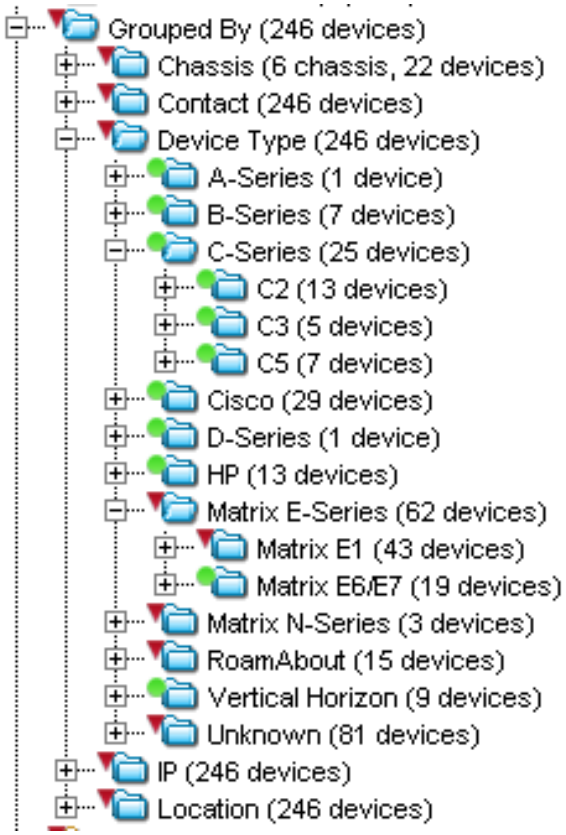


Figure 3.9 – Netsight hosts grouped by device type.

Following this structure and knowing beforehand that some models share similar command instructions and OID values, Zabbix would be populated with the following 11 host groups:

- Alcatel
- Cisco Catalyst

- Cisco NX
- Enterasys BCD
- Enterasys Matrix E1
- Enterasys Matrix E6
- Enterasys Matrix E7
- Enterasys VH
- HP
- TP-Link.

The creation of Host Groups in Zabbix is a simple task and is done by accessing the menu `Configuration>Host groups`, clicking on the `Create host group` button, inserting a name and clicking the `Add` button. Figure 3.10 illustrates this action.

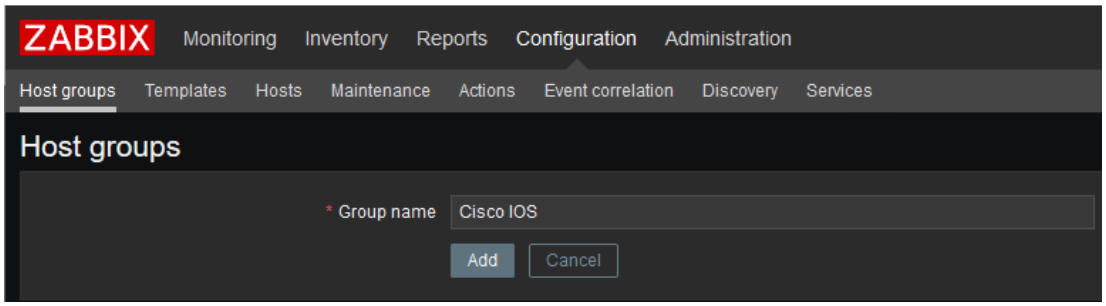


Figure 3.10 – Zabbix host group creation page.

3.4.3 Host discovery and templates

Templates are an important part of the Zabbix monitoring software. A template is a set of entities that can be conveniently applied to multiple hosts. The entities may be items, triggers, graphs, applications, screens, low-level discovery rules and web scenarios [29]. The use of templates facilitates the host creation process because a template can be applied to various hosts that share similar features, thus minimizing the workload. All the entities associated to a template are automatically applied to the hosts that share the same template and if something is changed in the template then all the hosts are affected by that change. Zabbix comes pre-configured

with diverse templates, some generic and others associated with manufactures, such as Cisco, HP, IBM, *etc*. The template feature is accessed through the `Configuration>Template` menu. Figure 3.11 shows the various Cisco templates that come pre-configures with Zabbix.

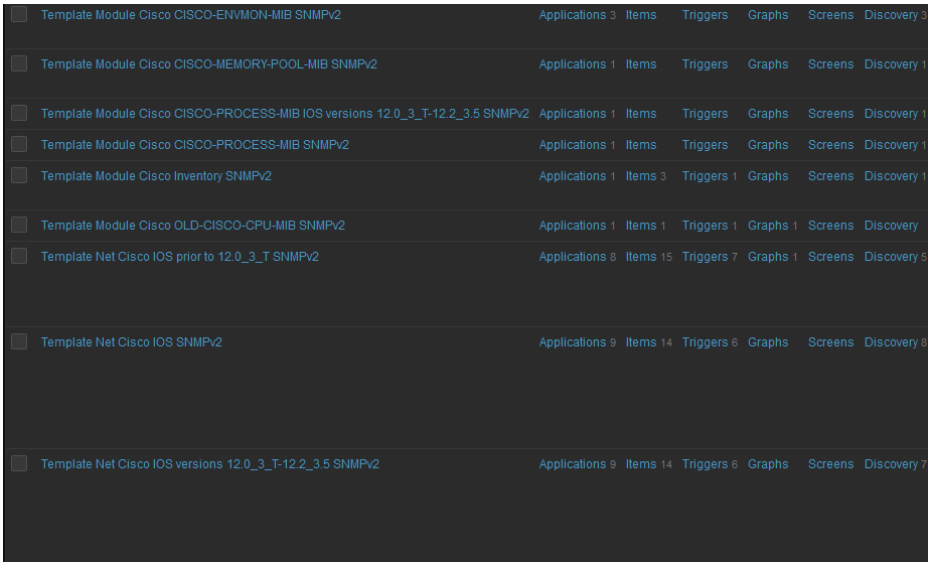


Figure 3.11 – Zabbix pre-configured cisco templates.

It was necessary to determine which network parameters would be monitored and if the existing templates were compatible or sufficient for the network devices. If the pre-configured templates were incompatible then it would be necessary to create new ones. The idea was to apply the same template (or templates) to all the hosts contained in the same host group, *e.g.*, the existence of an Enterasys E1 template for all Enterasys E1 devices contained in the Enterasys E1 host group. The templates would have to support the following network parameters described in Table 3.5.

Table 3.5 - Network parameters to monitor.

Application	Features
Network interfaces	bits received/sent, inbound/outbound packet errors/discards, speed, type, status
Status	device uptime, ICMP ping/loss/response time
Temperature	status, thresholds
Power Supply	status, installed
Memory	type, available, utilization
CPU	utilization
Fans	status
General	SNMP name/location/contact, description
Inventory	Hardware model/serial number, operating system

The parameters listed here are grouped by application, which is a Zabbix feature and is used to group items into logical groups. When configuring a host or template, an application can be created and assigned to a host item. This is a useful feature when monitoring items because it facilitates the viewing and filtering of the results.

The first step was to start adding the Cisco devices to Zabbix and apply the pre-configured Template Net Cisco IOS SNMPv2. This template contained practically every feature intended to be monitored. Adding hosts to Zabbix can be accomplished in two ways:

- Manually, by creating a new host
- Automatically, by using the discovery and action features.

Adding hosts manually to Zabbix is done by clicking the `Create host` button through the `Configuration>Hosts` menu and filling out the necessary host name (or IP address), host group, SNMP interface and linked templates fields. The downside of this method is the repetition necessary for each new host to be added, so using the automatic discovery process could easily streamline the whole process.

Zabbix offers an effective network discovery functionality, which periodically scans a defined IP range, and based on a set of defined checks, can generate a discovery event as for example a discovered host. Discovery events can be the basis of relevant actions, such as sending notifications, adding/removing hosts, adding hosts to a group and linking hosts to a template. These actions can be configured with certain conditions, such as a specific discover rule or a result from a discovery rule. Figures 3.12, 3.13 and 3.14 show an example of a discovery rule, an action and an action operation used to add Cisco hosts to Zabbix.

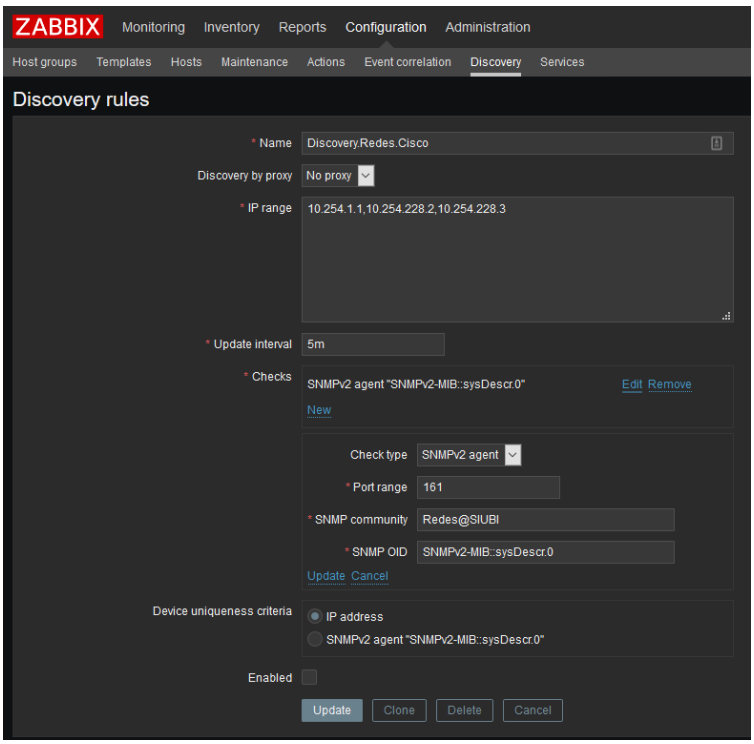


Figure 3.12 – Zabbix discovery rule configuration page.

This discovery rule was programmed with three Cisco 2960x management IP addresses, a check for the system description OID specified with the SNMPv2 community name and a uniqueness criteria by IP address. If a device with the same IP already exists, it will be considered already discovered and will not add that host.

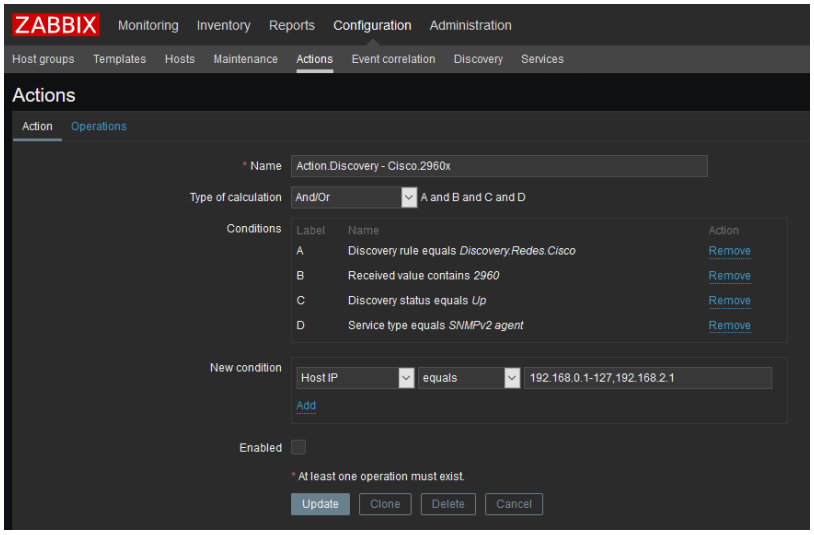


Figure 3.13 – Zabbix actions configuration page.

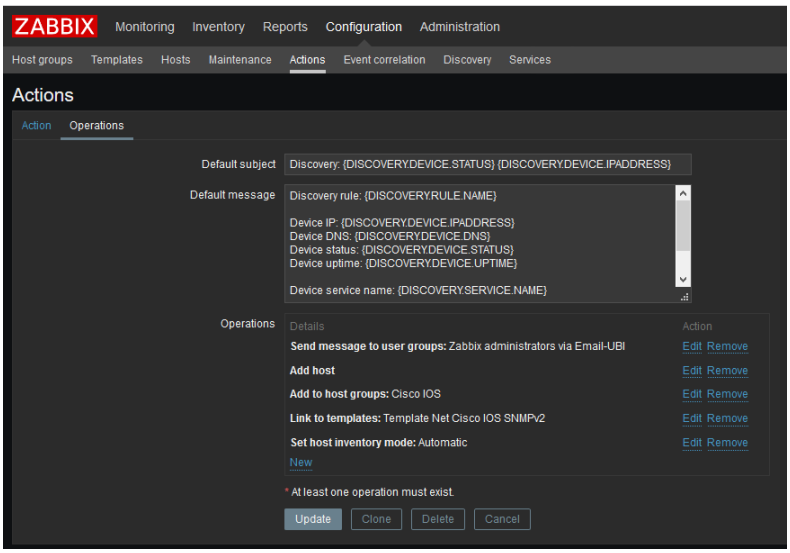


Figure 3.14 – Zabbix action operations page.

This action was specified with a few conditions: the previous Cisco discovery rule, the discovery check result has to include the string *2960*, the discovery rule and SNMP2 client agent have to be working.

The results of the action are defined in the operations section. In the case that the conditions are met, a message will be sent to the system administrator, the host is added to Zabbix and to the Cisco IOS host group, the host is linked to the Cisco pre-defined template and lastly the inventory mode is set to Automatic. The host inventory mode setting automatically fills

the host's inventory fields with relevant information such as device name, hardware, operating system, *etc*. For the send message operation to execute successfully, it was necessary to define a new Zabbix media type, specifying the SMTP server for sending email. This was achieved by clicking the `Create media type` button through the `Administration>Media types` menu and entering the information as shown in Figure 3.15.

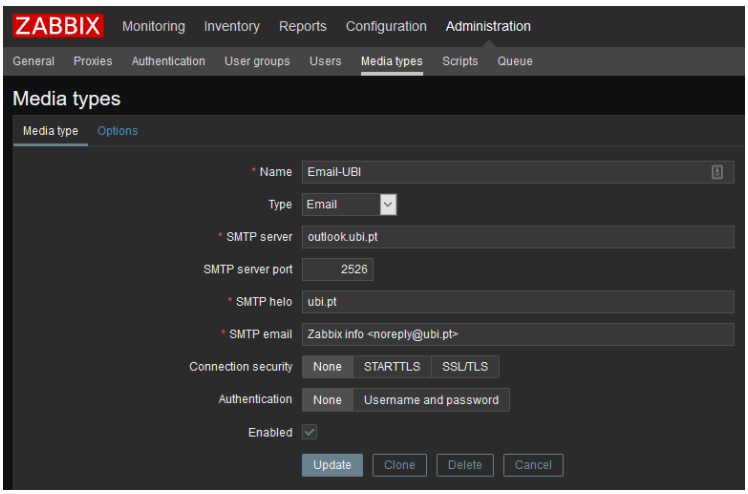


Figure 3.15 – Zabbix media type configuration page.

The send message operation would send a message to the Zabbix administrator group using the Email-UBI media type. The administrator group includes the users with administrative rights and is where the destination email is defined. Therefore, whenever a new host was found and successfully added to Zabbix, the administrator would receive an informational email. Figure 3.16 shows the users configuration page.

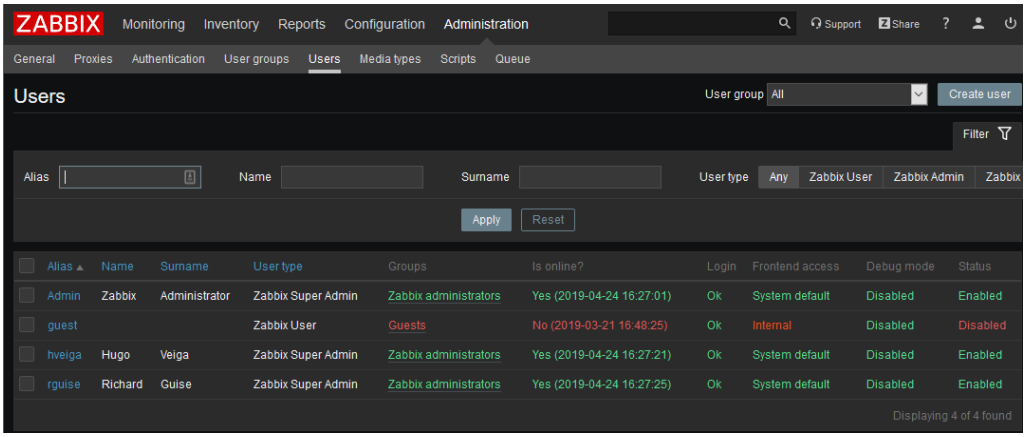


Figure 3.16 – Zabbix users configuration page.

As referred earlier, the template Net Cisco IOS SNMPv2 had every entity necessary for monitoring Cisco devices. This template consisted of a group of linked templates, each serving a specific purpose, such as, interface monitoring, CPU/memory/PSU/temperature/fan monitoring and host inventory.

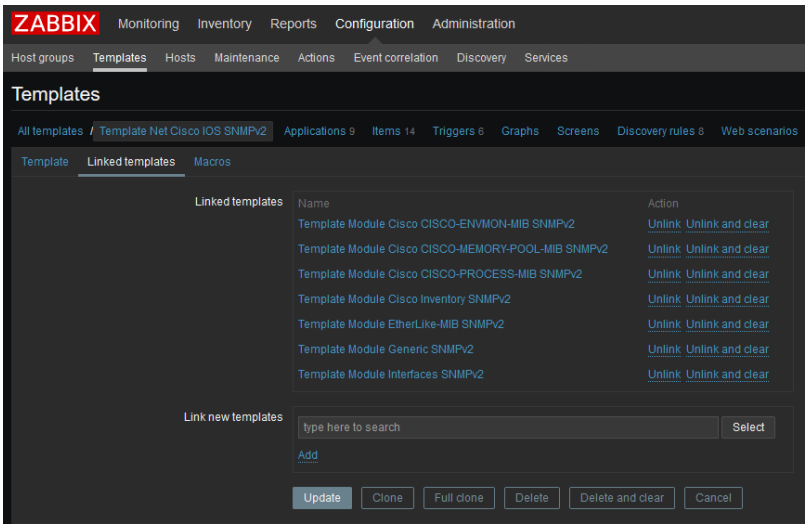


Figure 3.17 – Zabbix linked Cisco templates page.

The template also had a low-level discovery feature which provided a way to automatically create items, triggers, and graphs for different entities through the use of discovery rules and associated item/trigger/graph prototypes, removing the need to create items manually. For example, the discovery of all the network interfaces of a Cisco 2960x 48 port switch, was

accomplished with a Network Interfaces Discovery rule that indexed the device's various SNMP interface characteristics (description, type, status, name) with the use of macros, the interfaces OID's and the SNMP index. This information is then passed to the item, trigger and graph prototypes, automatically creating various entities each with its own name. Figures 3.18 and 3.19 shows an example of a low-level discovery configuration for the discovery rule and the item prototype.

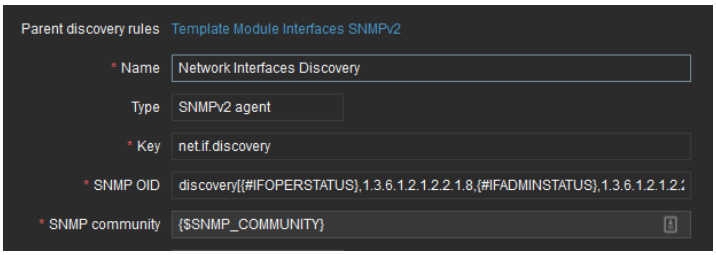


Figure 3.18 – Zabbix network interfaces discovery rule configuration page.

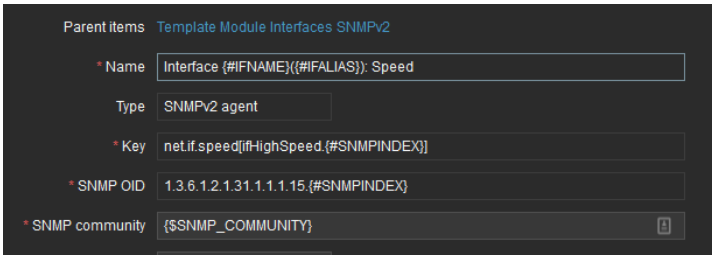


Figure 3.19 – Zabbix item prototype configuration page.

After the discovery process, the new host is added to the appropriate host group and linked to the necessary templates as defined in the action, thus creating all of the host's entities. The hosts are listed in `Configuration>Hosts` menu and are accessed by clicking on the host name. Figure 3.20 demonstrates the information that is viewed when clicking on the host name.

The figure shows the different types of entities, grouped by items, triggers and graphs, all created by the applied template and its low-level discovery process. The template also created application groups, which associate an item to a determined application, for example, a network interface

operational status item belonging to the `Network Interfaces` group or the system temperature belonging to the `Temperature` group. This allows the user to filter outputs by the type of application. There is a section for defining or altering `User Macros` and a section for configuring the `Host Inventory` fields that can be configured to populate automatically certain fields as for example: details of the type of device, serial number, location, responsible person, *etc*.

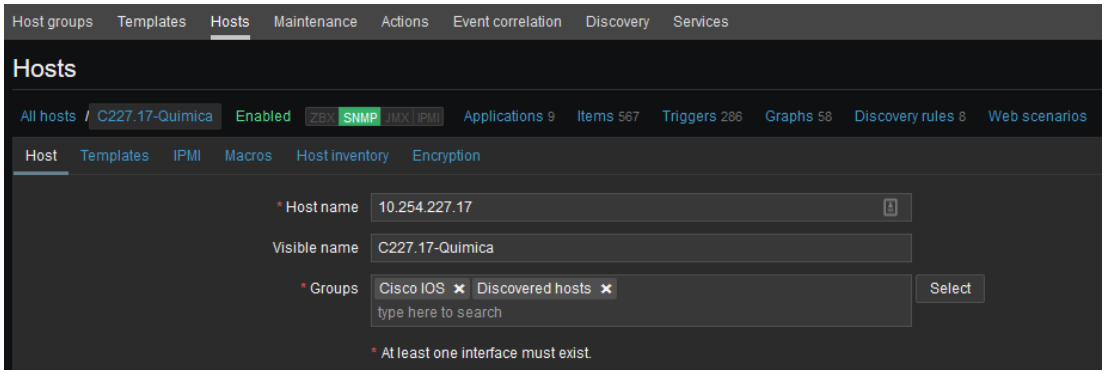


Figure 3.20 - Zabbix host configuration page.

By clicking on the items tab, Zabbix displays a list of all the host's monitored items as well as associated triggers, discovery time intervals and status, as shown in Figure 3.21.

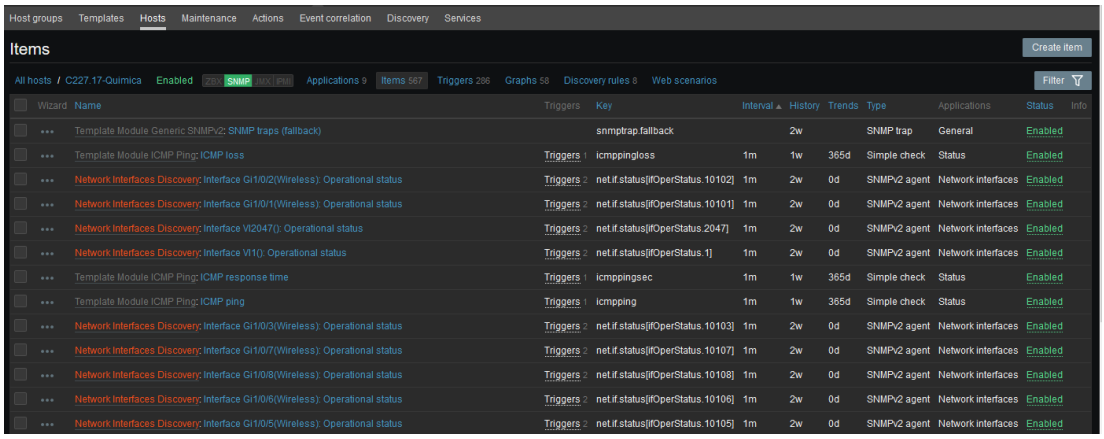


Figure 3.21 - Zabbix host items visualization page.

The item's properties are accessed by clicking on its name. Here it can be seen the SNMP OID used to access the device's information, the update

interval used to refresh the information and the application to which the item is associated. The preprocessing tab is used to define and execute transformation rules for the received item values. Figure 3.22 is an example of the properties of the item Interface `Gi1/0/2(Wireless): Operational status`.

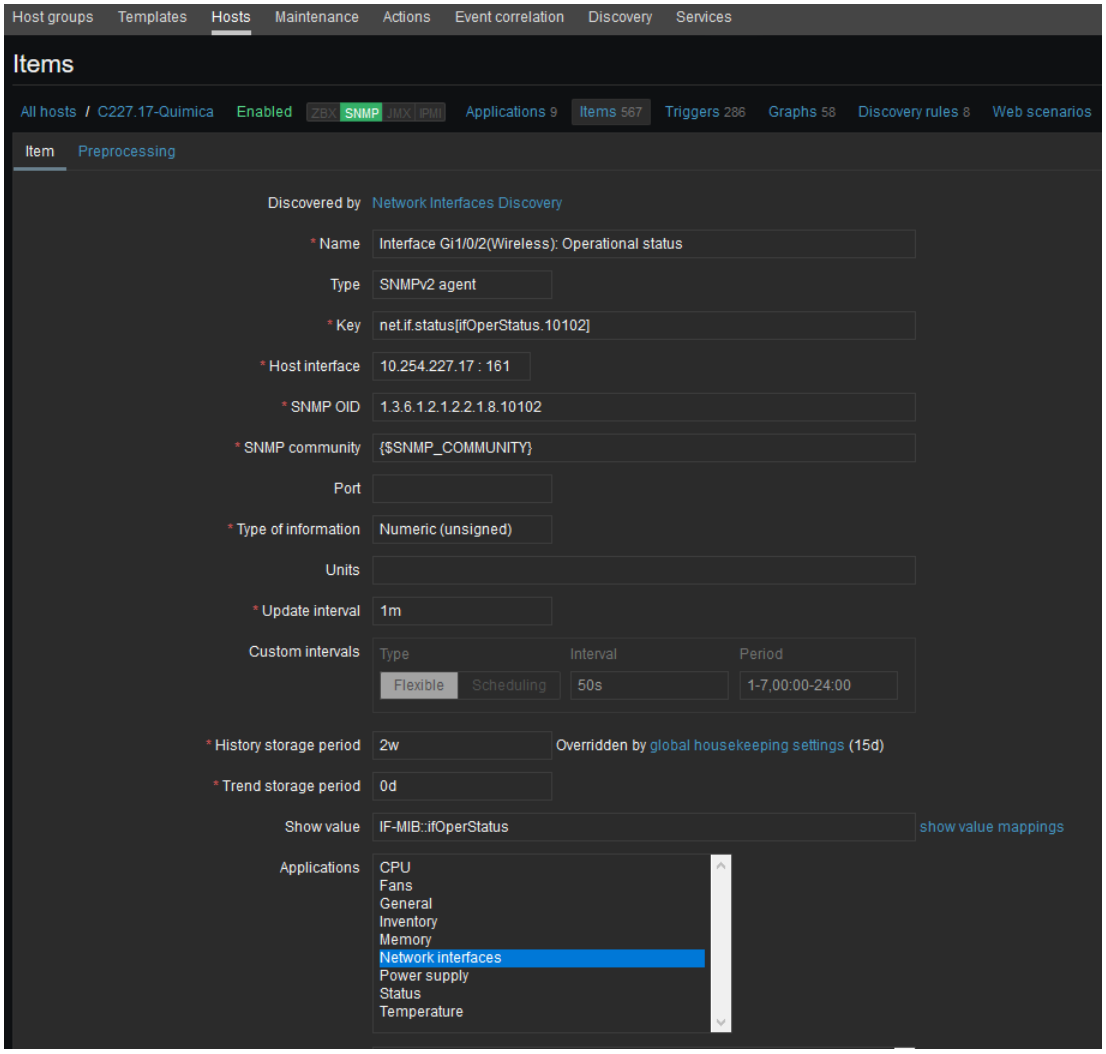


Figure 3.22 - Zabbix host items configuration page.

Figure 3.23 shows the Triggers page where the triggers are defined. Trigger expressions define a threshold of what state of data is acceptable. Therefore, should the incoming data surpass the acceptable state, a trigger is "fired" or a status change to `PROBLEM` occurs [30].

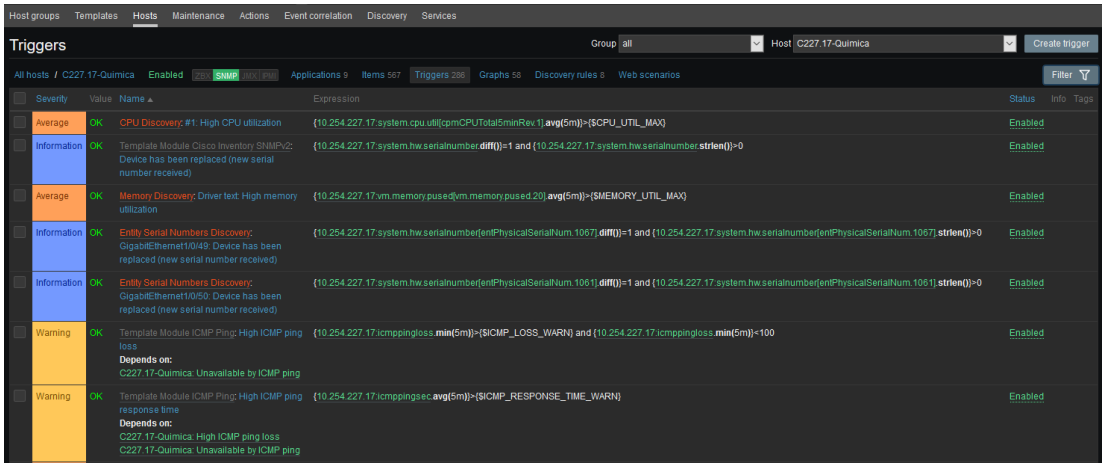


Figure 3.23 – Zabbix host triggers visualization page.

Figure 3.24 shows the Graphs tab containing all the graphs created for a particular host. A graph is a visual representation of the entity information, making it easier for analysis. Clicking on the graph name displays the graph properties.

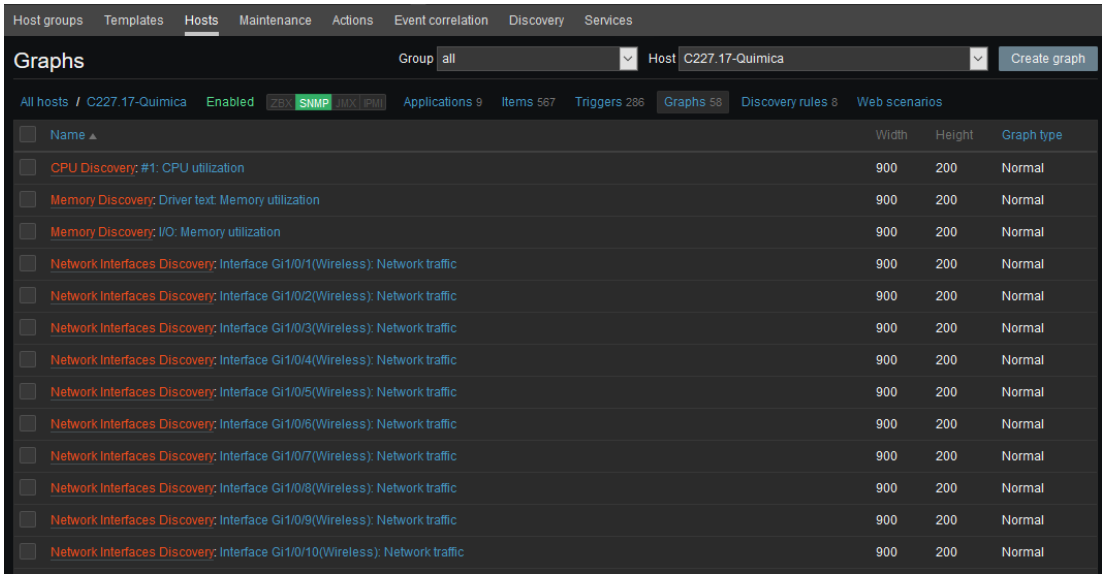


Figure 3.24 – Zabbix host graphs visualization page.

3.4.4 Creating templates

The existing Cisco templates served perfectly the monitoring specifications for UBI's Cisco devices, but Zabbix did not have specific templates for the Enterasys devices. Although there was one template for Extreme Networks (the company that acquired Enterasys in 2013), the problem was that the existing Extreme template was aimed at the EXOS operating system, which was not the case for UBI's Enterasys devices. The solution was to create new templates for the OIDs that where not supported by the existing templates, while also trying to utilize the more generic and compatible Zabbix templates.

The approach was to add one host from each type of device model to Zabbix, apply the Net Cisco IOS SNMPv2 template and observe the results in the `Hosts-Items` web page. If the status of a specific item was set to `Not supported` or if one of an Applications group did not contain items, then it was implied that the template's SNMP OIDs did not exist in the device. Figures 3.25 and 3.26 demonstrate these issues.

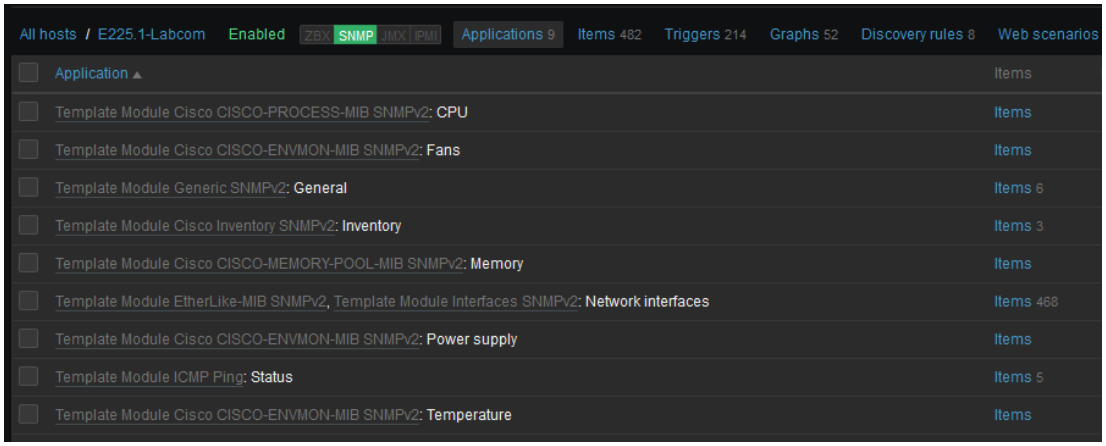


Figure 3.25 – Zabbix template applications page showing the absence of items.

In Figure 3.25, no items were created for the CPU, fans, power supply and temperature applications, which meant that the template OIDs where not compatible to the device's OID.

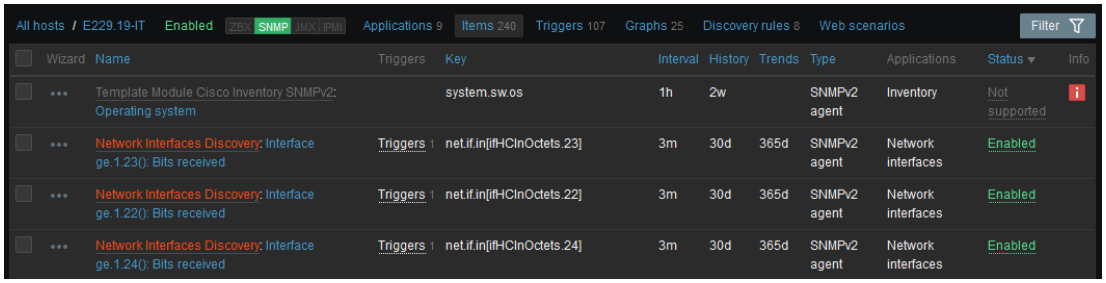


Figure 3.26 – Zabbix template items page showing a Not supported warning.

In Figure 3.26, it can be seen that the Operating system item from the Inventory application is not supported but the Network interfaces items are recognized.

This demonstrated the need for creating specific templates for the various Enterasys models. After observing the failed templates and items on all the different device models, it was concluded that the templates and nested templates needed, depending on the model and application, would be:

- Enterasys B and C series Template
    - Enterasys B and C series Inventory Template
    - Enterasys B and C series System Template
    - Module Generic SNMPv2 Template
    - Module Interfaces SNMPv2 Template
- Enterasys E1 series Template
    - Enterasys E1 series Inventory Template
    - Enterasys E1 series System Template
    - Module Generic SNMPv2 Template
    - Module Interfaces SNMPv2 Template
- Enterasys E6 series Template
    - Enterasys E6 series Inventory Template
    - Enterasys E6 series System Template
    - Module Generic SNMPv2 Template
    - Module Interfaces SNMPv2 Template
- Enterasys E7 series Template

    - Enterasys E7 series Inventory Template
    - Enterasys E7 series System Template
    - Module Generic SNMPv2 Template
    - Module Interfaces SNMPv2 Template
- Enterasys VH series Template
    - Enterasys SNMPv1 Interfaces Simple Template
    - Module Generic SNMPv1 Template.

Nesting is a way of one template encompassing one or more other templates. It makes sense to separate out individual templates entities for various services, applications, *etc*. resulting in the creation of quite a few templates all of which may need to be linked to quite a few hosts. To simplify the scenario, it is possible to link some templates together, in one "nested" template. The benefit of using nested templates is that the linking of only one template to a host will allow that host to inherit all entities of the linked templates automatically [31].

The Module Interfaces SNMPv2 template that came with Zabbix was compatible with every device series except the older VH model, which only accepted SNMPv1 protocol. This template is useful for monitoring the devices network interfaces traffic and status, as for example, bits received/sent, speed and operational status. The Module Generic SNMPv2 template was useful for showing the device's SNMP location, name, uptime, contact, *etc*. For the older VH models, Zabbix SNMPv1 Generic and interfaces Simple templates where sufficient for extracting the same information as the SNMPv2 devices.

It was decided that the creation of the new templates would fall into two categories: Inventory and System. The Inventory template would monitor such items as model name, serial number and operating system version. These items would also belong to the Inventory application. The items monitored by the System template would also be divided into three application categories: CPU, Memory and Power Supply. This template would

monitor such items as CPU loads, memory utilization, power supply and fan status. Normally, these features are redundant on a network device, for example, various power supplies and fans, CPU and memory monitoring in different time intervals, so when possible, and low-level discovery rule would be created to index these multiple item values.

The fundamental step for creating a template is figuring out the item's OID used to access the SNMP value on the device. One easy way to do this is accessing the CLI of the network device and access the information that is to be monitored. Then, by means of a SNMP MIB browser software, query the device's MIB database and try to find that same information. If found, then use that OID in the Zabbix template. Figures 3.27 and 3.28 demonstrate this process and more specifically, the creation of a template item used to monitor the available flash memory of an Enterasys switch series B.

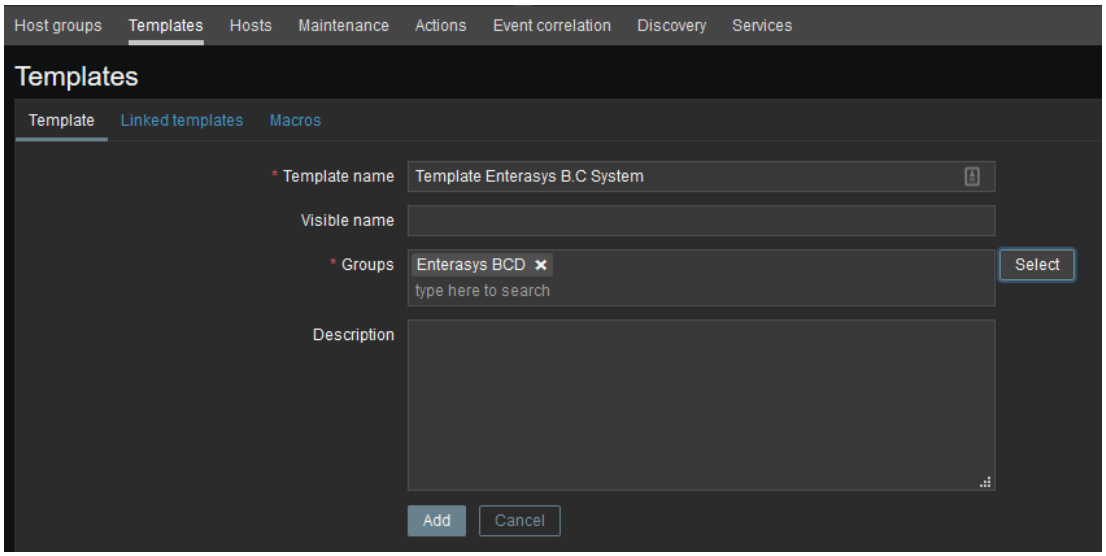


Figure 3.27 – Zabbix template creation configuration page.

The template creation is achieved through the `Configuration>Templates` menu and clicking on the `Create template` button. It is necessary to give a name and assign a host group. There is also a linked templates option, which allows the nesting of one or more templates to the parent template, and a Macros option to define new macros or overload existing ones.

After creating the template, new items can be assigned through the `Items` tab and `Create item` button. A name and a key must be assigned to unequivocally identify the item. The SNMP OID and community fields are necessary in order to access and locate the object ID in the network device. There is also a field to specify the Application to which the item belongs and a preprocessing tab which allows the definition and execution of transformation rules for the received item value.

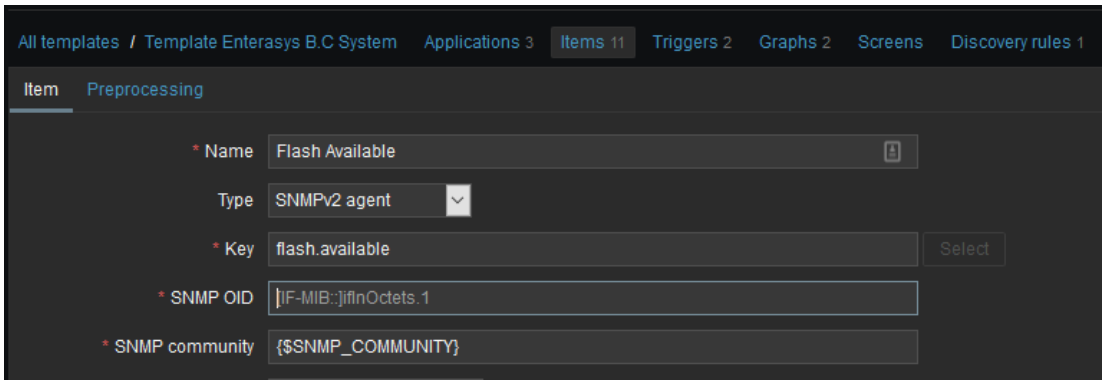


Figure 3.28 – Zabbix template item configuration page.

The method used to discover the entity OID was to query the device's MIB with the iReasoning MIB Browser software [32] and search for a specific value or string that was allusive to the item.

```
Storage Utilization:
Type    Description                Size(Kb)       Available (Kb)
-------------------------------------------------------------
RAM     RAM device                   262140               73852
Flash   Images, Config, Other        125038              106800
E232.2-Sotao(su)->
```

Figure 3.29 – Telnet session to an Enterasys switch.

Figure 3.29 shows the result of the CLI command *show system utilization* given on an Enterasys B series switch. Searching for the string *flash* in the iReasoning software returns the `ResourcesStorageTable` MIB where the OID for the available flash item can be found. Figure 3.30 shows the iReasoning software main screen.

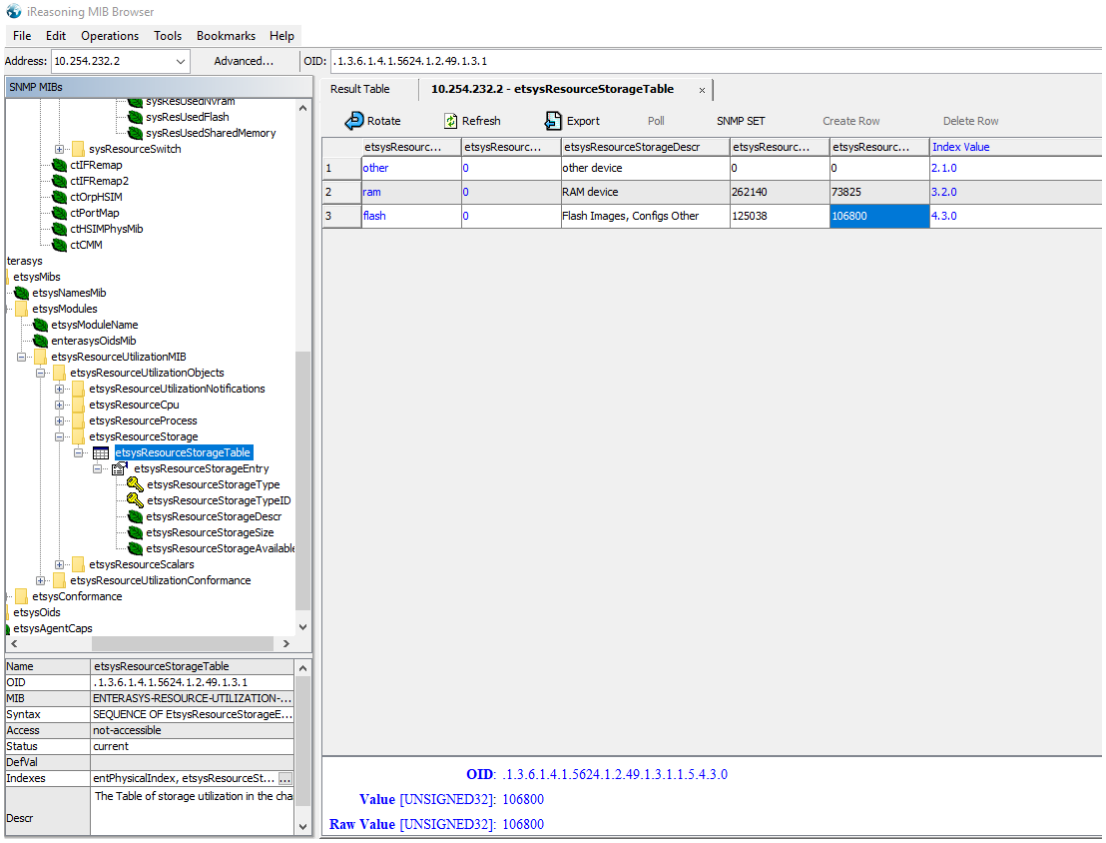


Figure 3.30 - iReasoning software showing a device's MIB.

The last step was to insert the discovered OID value into the Zabbix item SNMP OID field and check if the item status was set to enabled as shown in Figure 3.31.

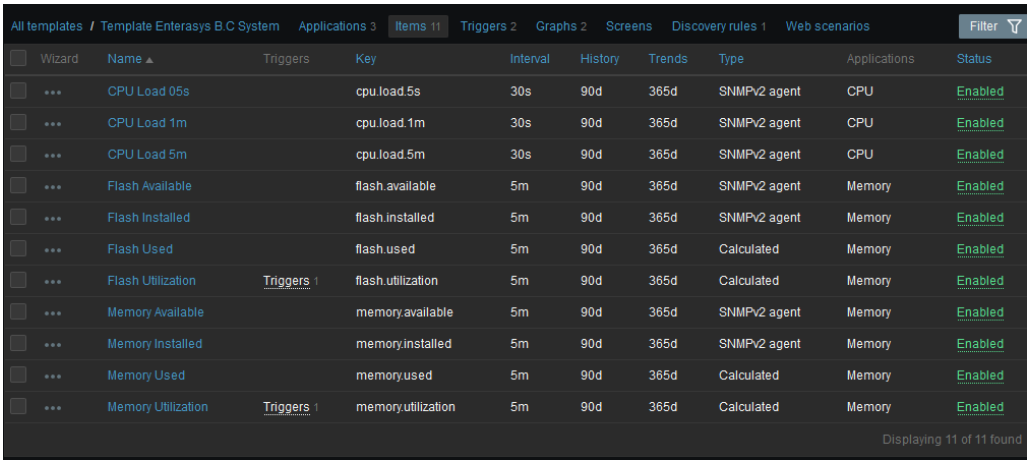


Figure 3.31 - Zabbix template items status page.

This procedure was replicated for different item OIDs to support the monitoring needs: CPU loads for various intervals (5 seconds, 1 and 5 minutes), installed and available memory. Zabbix offers the possibility to create item based on calculations over existing item. Enterasys devices did not have an OID for memory utilization, so one was calculated by dividing the used memory item by the installed memory item and multiplying by 100 to achieve a percentage value. The used memory item was also a calculated item achieved by subtracting the installed memory with the available memory, as shown in Figures 3.32 and 3.33.

The creation of triggers were also necessary in order to evaluate the received data received and alert in case of an issue. It was necessary to create triggers for high ram and flash memory usage and inoperable power supplies. The existing generic templates already covered device status and network interfaces. Creating a trigger is accomplished through the `Create trigger` button in the `Triggers` tab of the template that is being programmed. The name and expression field are mandatory and a severity type field can be chosen.

Figure 3.34 demonstrates the expression used to evaluate the system memory utilization data. The trigger uses the `Memory Utilization` calculated item described previously. Zabbix has an expression constructor that facilitates the creation of the expression, which uses the average of the last five values taken. The template macro `$MEMORY_UTIL_MAX` was also created to contain the threshold value for which the trigger should fire, which in this case was 90%. So if the average of the last five values is over 90% then the trigger fires. A similar trigger was also created for high flash utilization.

Zabbix also offers a visual representation of the data with the use of graphs. Graphs are a powerful feature that allows the grasp of the data flow at a glance, correlate problems, discover when something started or make a presentation of when something might turn into a problem [33]. It was

necessary to create graphs for the memory and flash monitoring. This was accomplished by accessing the template's Graphs tab and clicking on the `Create graph` button. The mandatory fields are the graph name, width, height and visualized items as shown in Figure 3.35.

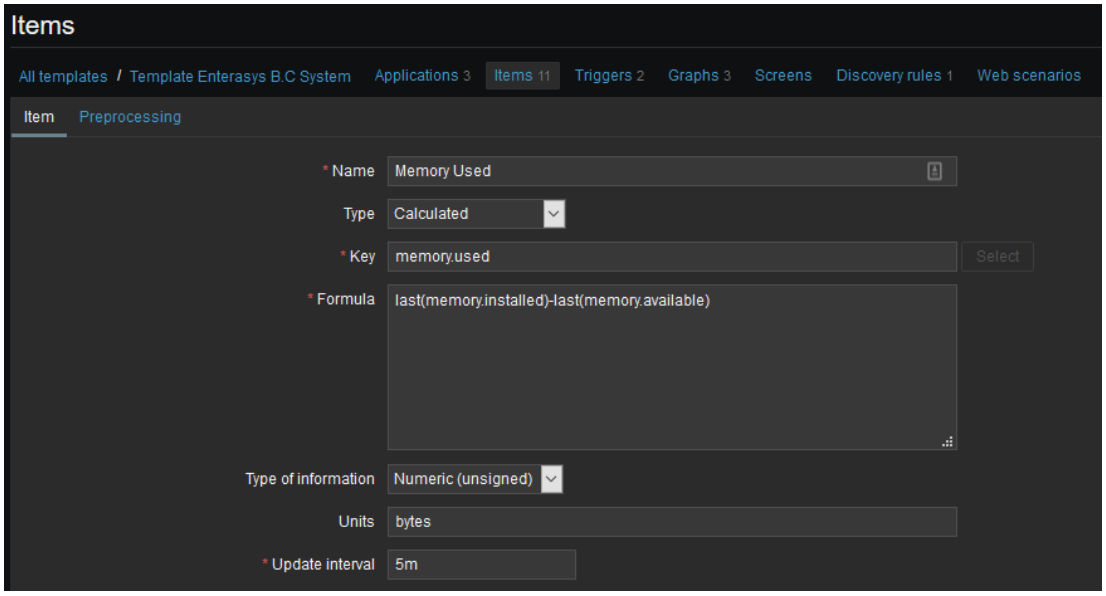


Figure 3.32 - Zabbix `Memory Used` calculated item configuration.

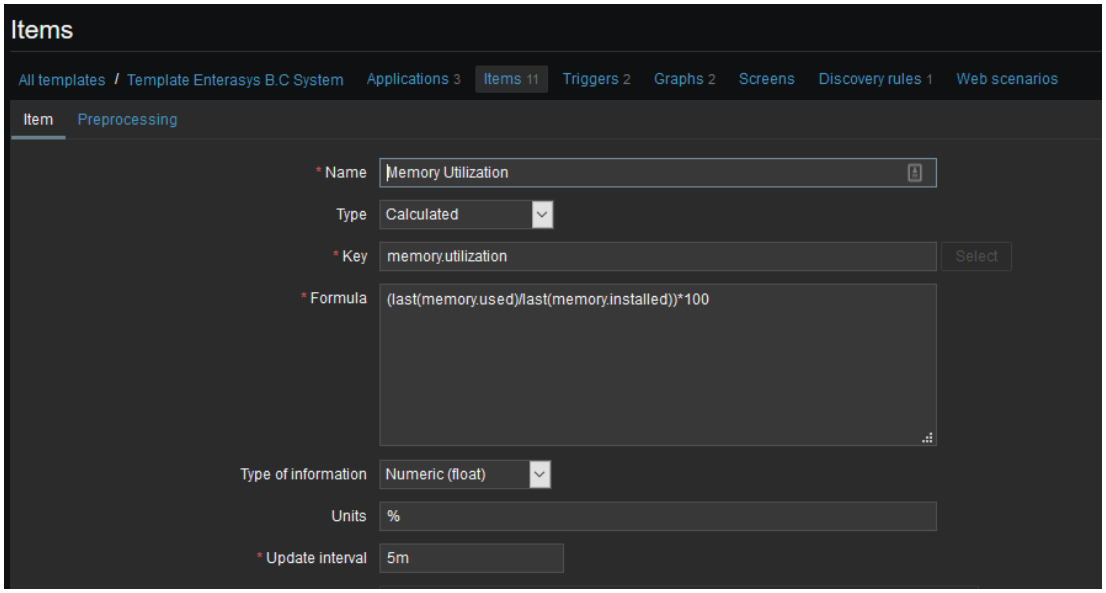


Figure 3.33 - Zabbix `Memory Utilization` calculated item configuration.

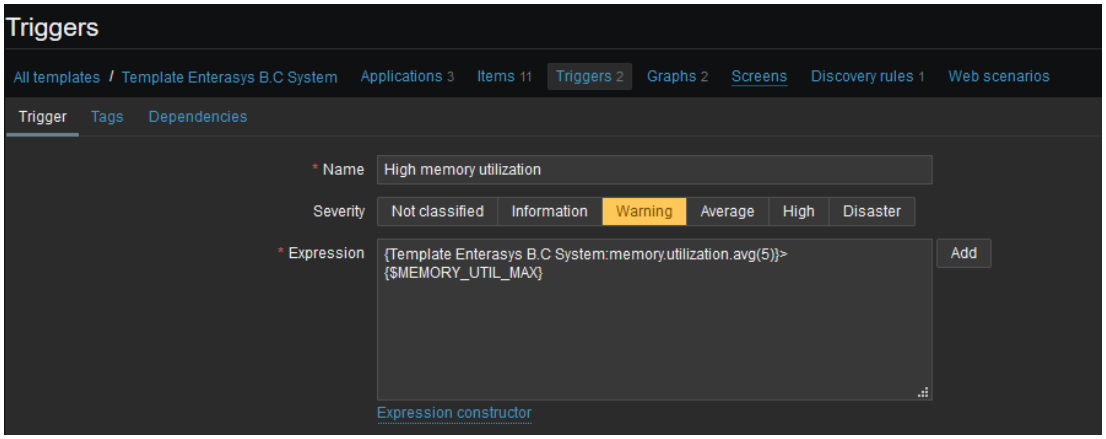


Figure 3.34 - Zabbix `High memory` trigger configuration.

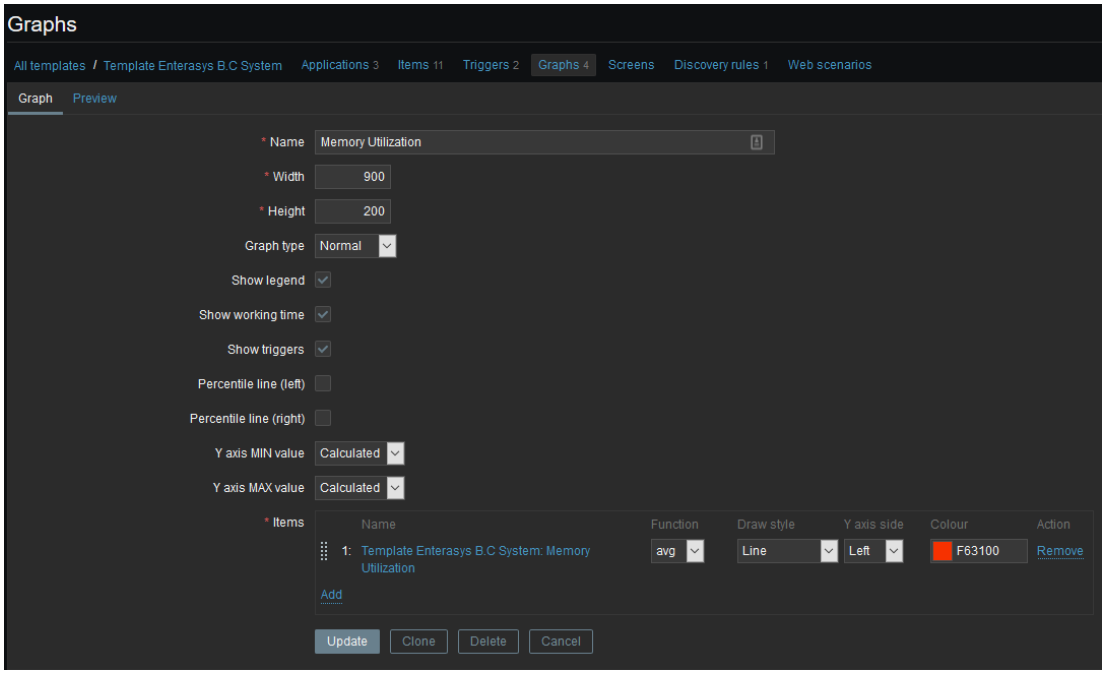


Figure 3.35 - Zabbix `Memory Utilization` graph creation.

One last item in need for monitoring was the power supply unit. Since a device can have more than one PSU, it was necessary to create a low-level discovery rule to dynamically index the various units. A discovery rule can be created through the template's `Discover rule` tab and by clicking on the `Create discovery rule` button as shown in Figure 3.36.

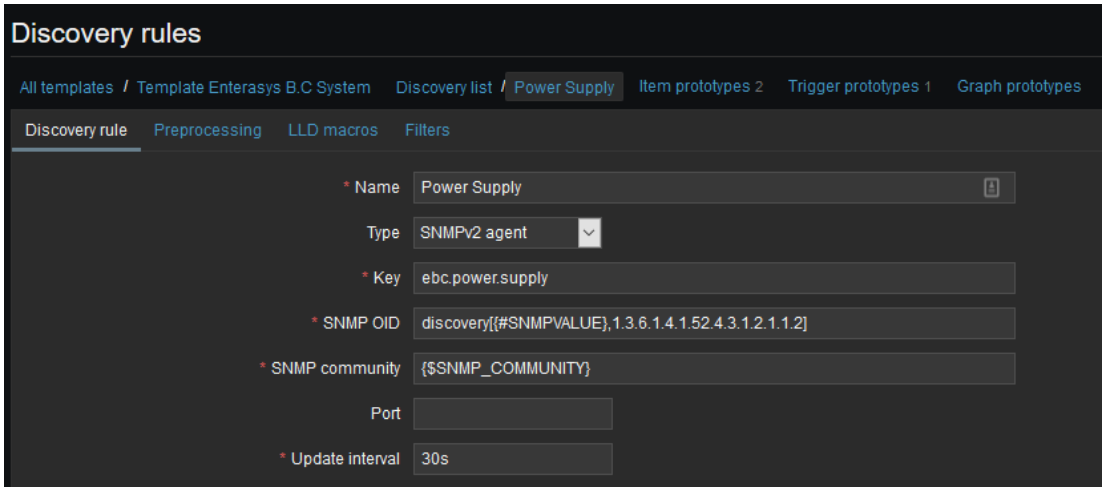


Figure 3.36 – Zabbix `Power Supply` discovery rule configuration.

The SNMP OID used to create this rule, should be one that encompasses all the OIDs for each existing PSU. For example, running the `snmpwalk` command for an Enterasys B switch with the OID 1.3.6.1.4.1.52.4.3.1.2.1.1.2, returns two OIDs, one for each of the switch's PSU (see Figure 3.37). The low-level discovery feature will index these results with the pre-defined `#SNMPINDEX` macro (in this example it will record the indexes 101 and 102) which can later be used when creating the item, trigger and graph prototypes.

```
[root@zabbix ~]# snmpwalk -v2c -c Redes@SIUBI 10.254.227.6  1.3.6.1.4.1.52.4.3.1.2.1.1.2
SNMPv2-SMI::enterprises.52.4.3.1.2.1.1.2.101 = INTEGER: 3
SNMPv2-SMI::enterprises.52.4.3.1.2.1.1.2.102 = INTEGER: 4
[root@zabbix ~]#
```

Figure 3.37 – snmpwalk command for discovering a device OID.

After the creation of Discovery rule, it was necessary to create an item prototype for the PSU status and redundancy monitoring. A look at the Enterasys Power Supply MIB file showed the needed OID values and the different sates these items could have.

```
ctChasPowerSupplyState=1.3.6.1.4.1.52.4.3.1.2.1.1.2 {
      infoNotAvailable ( 1 ),
      notInstalled ( 2 ),
      installedAndOperating ( 3 ),
      installedAndNotOperating ( 4 )}
ctChasPowerSupplyRedundancy=1.3.6.1.4.1.52.4.3.1.2.1.1.4 {
      redundant ( 1 ),
      notRedundant ( 2 ),
      notSupported ( 3 )}
```

Before creating the items, a quick test was made by `SNMP walking` the device to obtain these values but it was realized that the results were in number form, making it difficult to understand the results. Fortunately, Zabbix offers a `Value Mapping` feature that allows a mapping of a value to a text, so that the user can view these results in a more friendly presentation. This configuration is accessible by choosing the `Value Mapping` option in the drop-down list through the `Administration>General` menu. A value mapping was created for each item as seen in Figure 3.38.

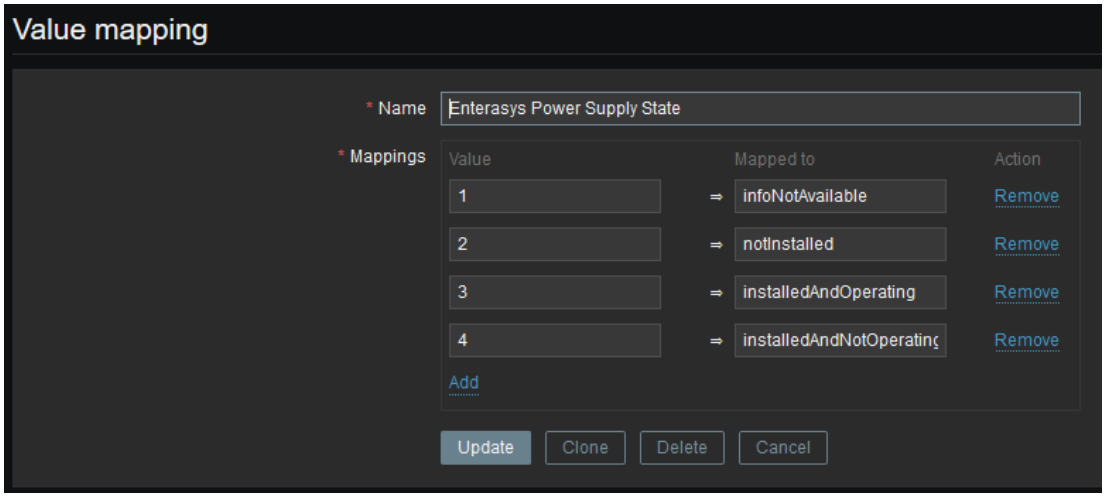


Figure 3.38 - Zabbix value mapping configuration page.

The next step was to create the items, which is done by clicking on the `Create item prototype` button in the `Item protoypes` tab of the

Discover rule. Concatenating the `#SNMPINDEX` macro to the object name and OID enables the creation of multiple PSU items (one for each of the device's PSU) and each with its own different OID. The association of the value mapping with the item is done in the `Show value` field. Figure 3.39 demonstrates this configuration.

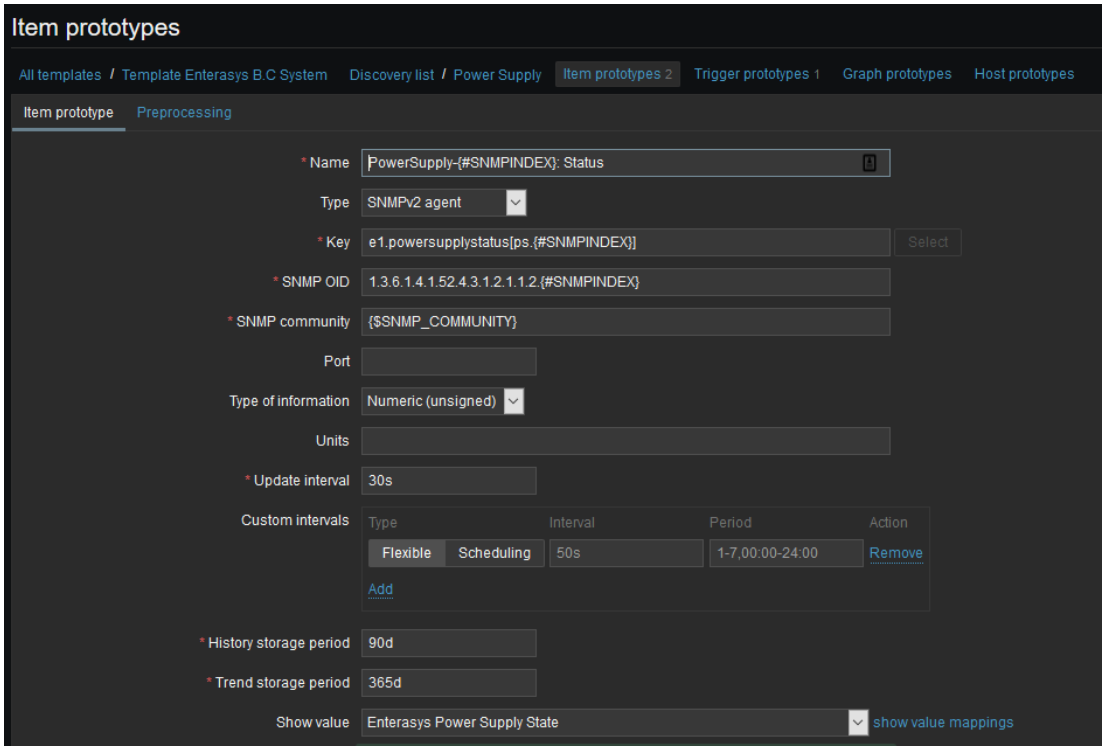


Figure 3.39 – Zabbix item prototype configuration page.

It was also necessary to create a trigger prototype for the PSUs. This was accomplished in the `Trigger prototypes` tab with the `Create trigger prototype` button. The configuration is similar to the item prototype in the way that the `#SNMPINDEX` macro is used. It was necessary to be alerted if a power supply unit was down and understand if the device had redundant power supplies.

It was necessary to create a few scenarios as to understand the combination of SNMP values that could exist with a non-operating PSU. While some devices had redundant PSUs, others would have only one. The redundant PSUs were shut down to analyze the resulting SNMP MIB and understand the various

combinations between the SNMP state and redundancy values. In the case of only one PSU, the SNMP MIB would strangely show entries for a non-existing second PSU: `installedAndNotOperating(4)` state and `notSupported(3)` redundancy. This combination of values had to be avoided to not provoke false positives. After careful analysis, it was concluded that the correct combination of SNMP state with redundancy in regards to a PSU being down would be `installedAndNotOperating(4)` state with `notRedundant(2)` redundancy. Figure 3.40 shows the configuration of the power supply trigger prototype.

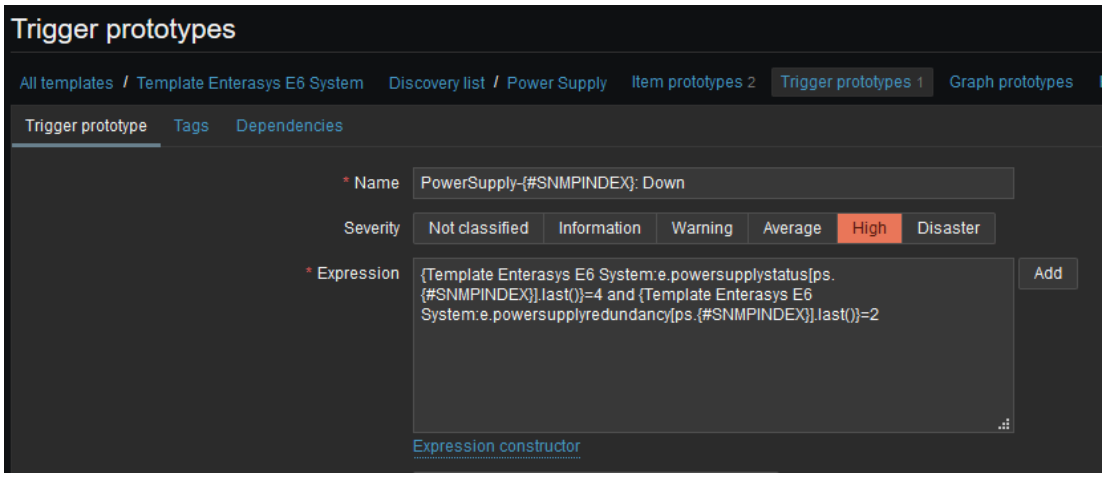


Figure 3.40 – Zabbix trigger prototype configuration page.

Once the basic Inventory and System templates were created and since they were compatible with practically every model except the older VH models, there templates were replicated for each type of model. This would allow future changes to the template if something more specific would be needed to monitor that wasn't available on all he devices. Zabbix offers a full clone feature that duplicates the template that only requires a different template name. The process was to duplicate the template, assign a different name and group and use that specific template when discovering new hosts. Figure 3.41 shows the summary of the new Enterasys templates and the linked hosts.

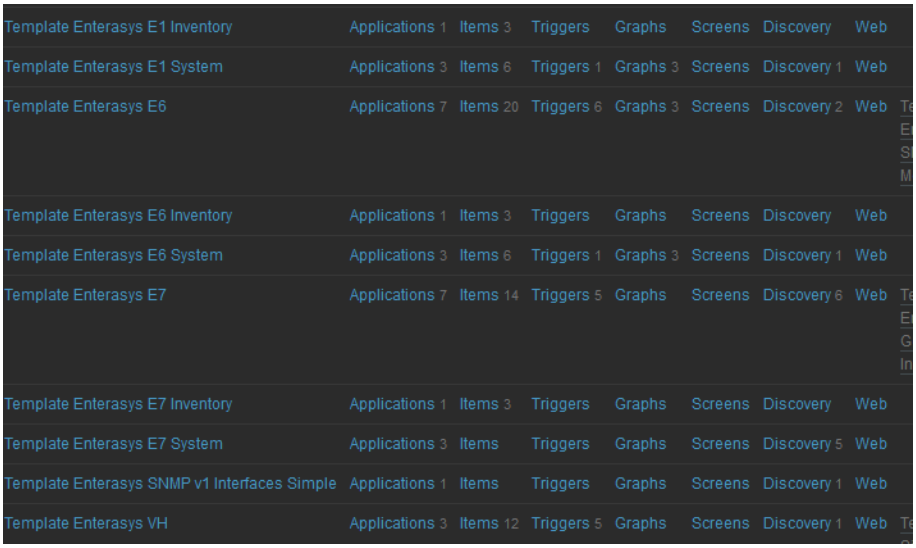


Figure 3.41 – Zabbix Enterasys templates page.

Once all the Cisco and Enterasys devices were accounted for in terms of template attribution, it was time to focus on the remaining vendors, such as Alcatel, HP and TP-Link. Zabbix contained a Net HP Enterprise Switch SNMPv2 template, which proved sufficient for the monitoring needs. For the Alcatel and TP-Link devices, the included Net Cisco IOS SNMPv2 template was also sufficient.

3.4.5 API

When Zabbix discovers a host, the host name field is set with the device's IP address, making it difficult to identify each equipment. The monitoring would be facilitated if the devices were identified by their SNMP system name. Zabbix offers a second field named `Visible name` which substitutes the `Host name` field in the monitoring web pages. The problem was that it was not possible to automatically fill this field with the device's SNMP system name when the device was added to Zabbix. The hosts have an Inventory section, populated by some of its template items, which show relevant information including the system name, as seen in Figure 3.42.

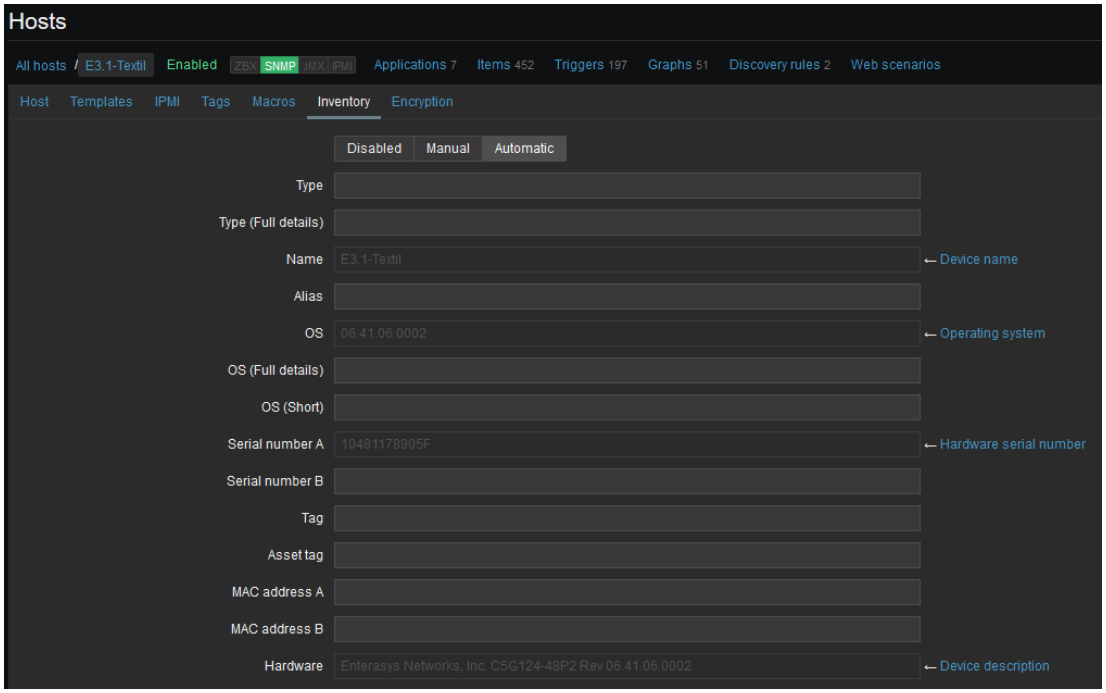


Figure 3.42 – Zabbix device inventory configuration page.

Figure 3.43 shows the field from the Device Name item of the Module Generic SNMPv2 template, which populates the name field of the device's Inventory section.

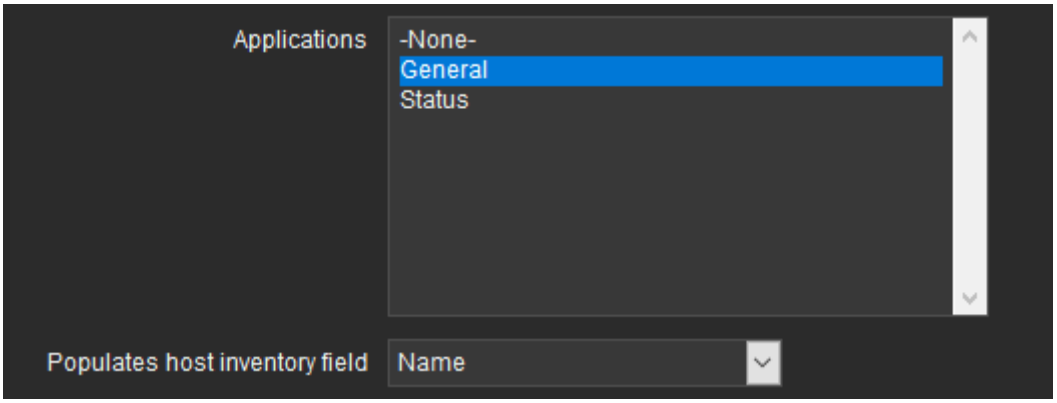


Figure 3.43 - Zabbix template item device name field.

The challenge now was to find a way to automatically take the Inventory Device name data and use it to populate the host Visible name field. It would also be useful if any change to the device system name could automatically be reflected in Zabbix. Fortunately, Zabbix has a web based API which allows

to programmatically retrieve and modify its configuration and provide access to historical data. Therefore, the idea was to create a Python script that could access the needed data and update the host visible name field in case it was different. This script could also be scheduled to run periodically to automatically check for changes in the device name. The code used to implement this task was the following:

```
#!/usr/bin/python
from pyzabbix import ZabbixAPI
user="Admin"
secret="xxxxxx"
zapi=ZabbixAPI("http://127.0.0.1/zabbix")
zapi.session.auth=(user, secret)
zapi.login(user, secret)
print("Connected to Zabbix API Version %s" %
zapi.api_version())
hosts=zapi.host.get(output="extend",selectInventory="extend")

if not hosts:
    print ("Empty list.")
    exit()


for h in hosts:
    name=(h['name'])
    id=(h['hostid'])
    inventory=(h['inventory'])

    if 'name' in inventory:

        if inventory['name'] != '':
invName=inventory['name'].replace(".redes.ubi.pt","")

            if name != invName:
                zapi.host.update(hostid=id,name=invName)
                print("ID: {}, Name: {}, Inventory:
{}".format(id,name,invName))
```

After the script was run, all the hosts visible name fields were updated successfully as shown in the Figure 3.44.

Figure 3.44 – Zabbix device host name field.

3.5. Conclusion

This chapter has described, in a detailed manner according to the best practices of the Network Administrator community, the steps taken to install, and configure Zabbix, including the steps taken to normalize the names and network environments of the devices of UBI's network.

The next chapter will focus on the results obtained from the monitoring system and some issues that needed to be addressed.

Chapter 4

4. Evaluation and tests

After completing the installation and configuration procedures as described in the previous chapter, it was time to start observing the Zabbix monitoring data. Some interesting and alarming events became instantly known once Zabbix started to compile the received host data. Statistics and trends in data also started to appear after a couple of days. Important issues regarding performance and excessive data would need to be addressed urgently.

4.1. Monitoring Issues

Once Zabbix became populated by hosts, several issues started to arise. The first issue was the multiplicity of device network interfaces that did not need monitoring, *e.g.*, console ports, management ports, stack ports, back planes *etc*. These superfluous interfaces would need to be removed from the system to facilitate the monitor viewing and to reduce system overhead. Zabbix offers a result-filtering feature that in conjunction with regular expressions can remove unwanted items from monitoring. The first step was to add some new regular expressions that matched the unwanted interface names to the already existing list of filtered expressions. This was accomplished through the `Regular expressions` option in the drop-down list of the `Administration>General` page. After careful inspection of all the network interfaces of each device type in the `Monitoring>Latest Data` page, it was concluded that the interface names that did not need monitoring were `console`, `StackPort`, `StackSub`, `host`, `mgmt0`, `bp`, `smb`, `lag`, `rtr` and `com`. These interfaces would have to be matched by regular expressions inserted in the `Network interfaces for discovery` list as shown in Figure 4.1.

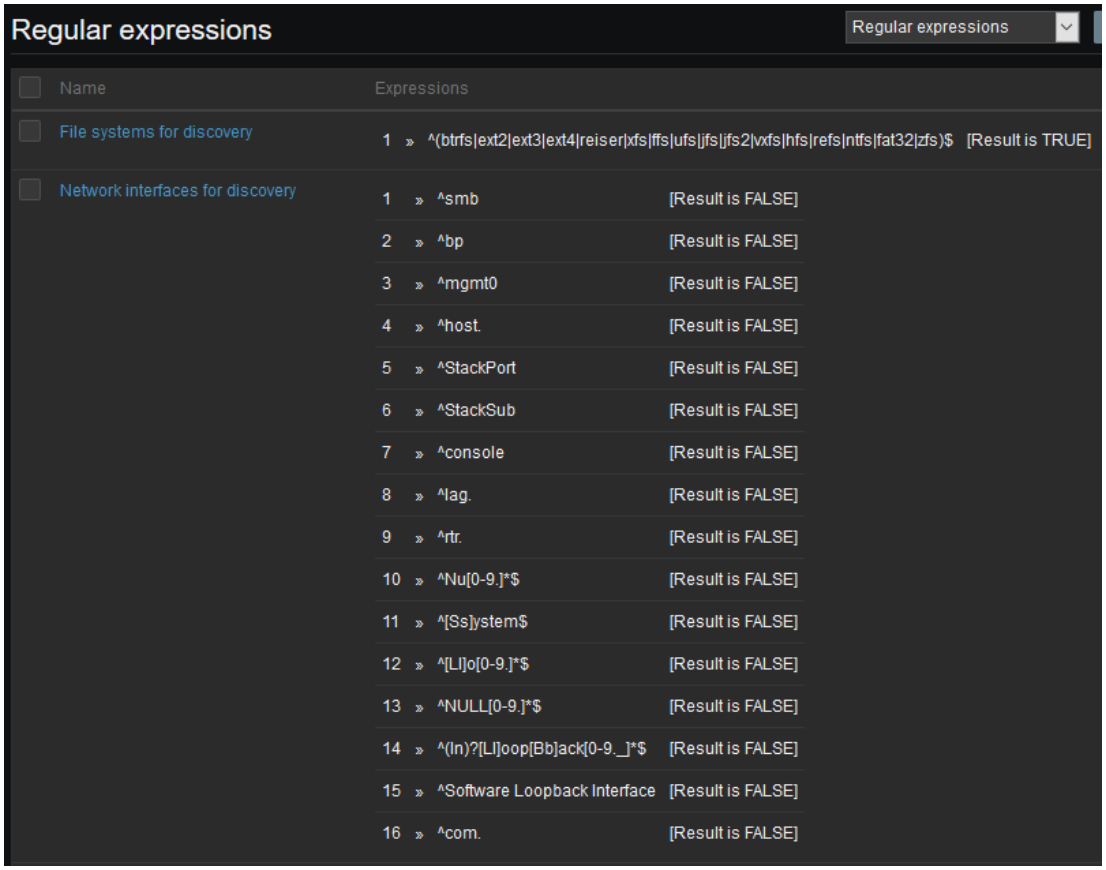


Figure 4.1 - Zabbix regular expressions configuration page.

All of the Network Interfaces Discovery rules, available in the Module Interfaces SNMPv2 template utilize the above list. Therefore, when a new host is discovered, the filter action will disregard any interface name that matches any of the above expressions. After this change, Zabbix stopped monitoring the unwanted interfaces. Figure 4.2 demonstrates the association of the filter rule with the discovery rule.

After filtering the unwanted interfaces, the main monitoring screen was still showing much unwanted information, mainly about device interfaces going down or changing to a lower speed. This was not important information, because it just showed that the end devices were powering down and that crucial information was being lost in all this noise. The interfaces that really mattered were the uplinks that interconnected network devices and not the user end devices. The important end devices that mattered were wireless access points, printers, servers, IP phones, access controls and IP cameras.

It is necessary to differentiate which interfaces were important for monitoring and which could be excluded, but this will be discussed in the following conclusions chapter, future work section.

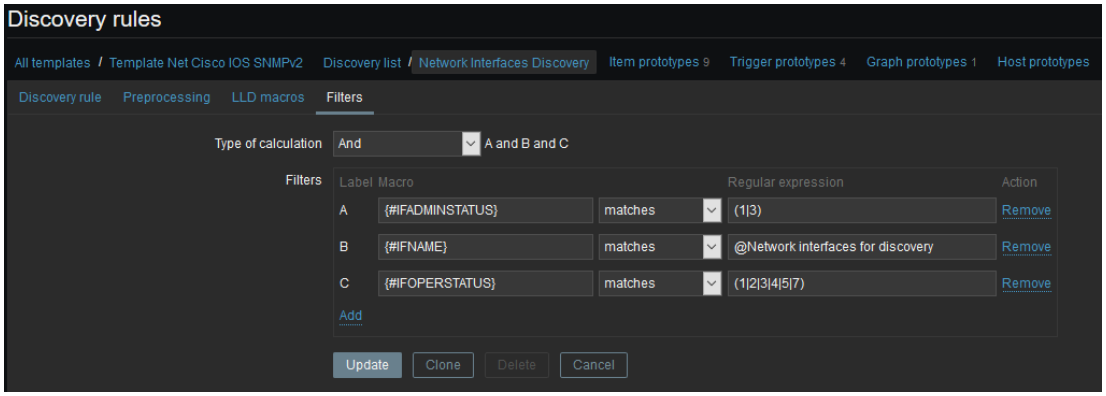


Figure 4.2 - Zabbix template discovery filters page.

4.2. Performance Issues

Zabbix started crashing after it became fully populated with 150 hosts. The `zabbix_server.log` file was showing out of memory errors in regards to the `CacheSize`, `TrendCacheSize`, `HistoryCacheSize` and `ValueCacheSize` parameters. Fortunately, the problems stopped after these values were incremented in the `zabbix_server.conf` file, shown below.

```
# joe /etc/zabbix/zabbix_server.conf
# CacheSize=8M
CacheSize=128M
# TrendCacheSize=4M
TrendCacheSize=24M
# ValueCacheSize=8M
ValueCacheSize=128M
# HistoryCacheSize=16M
HistoryCacheSize=80M
```

A grave issue occurred in Zabbix after a month of collecting monitoring data. The Zabbix database was becoming extremely large, around 100GB, mainly due to the ibdata1 file. It was quickly realized that the culprit was the Zabbix history data that was being kept. In the `Zabbix Housekeeping` menu, the default value of 365 days of saved history data was set, but it would have to be changed to a much smaller value of 15 days. This was enough because Zabbix also has a Trends function, which is a built-in historical data reduction mechanism. Trends stores minimum, maximum, average and the total number of values per every hour for numeric data types thus removing the need of storing a lot of history. Zabbix has a housekeeping process, which removes older data for a predefined storage period. This feature is accessible through `Administration>General>Housekeeping` menu, shown in Figure 4.3.

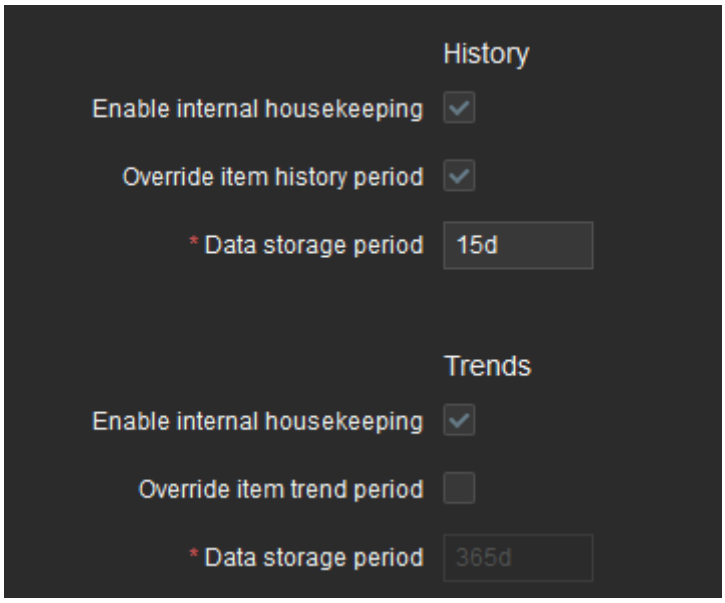


Figure 4.3 – Zabbix housekeeping configuration page.

Altering the default history value did not reduce the database size because there was no way to reduce the `ibdata1` file mainly due to the way the database was created. The `ibdata1` file contains all the tables and indexes of the MariaDB/MySQL database, so it is not possible to access the database tables separately to perform maintenance tasks. Zabbix keeps the history

data in the `history_uint` table, but because the database was not created with `innodb_file_per_table` enabled, the tables were not stored in their own tablespace or one file per table, resulting in the impossibility of reducing the `ibdata1` file. It was an oversight not enabling this feature, because the Zabbix guides clearly state that InnoDB should be enabled.

There were two options to fix this issue, the first was to backup the database, delete it and do a fresh install with `innodb_file_per_table` enabled and import the data again. After that, the history table could be truncated resulting in the shrinkage of the `history_uint` file. The problem with this option was that that truncating Zabbix tables is not recommended because it could create orphaned data in other tables. The second approach was to back up all the `hosts`, `templates`, `valuemaps`, `regex` and everything else that was configured through the GUI, reinstall Zabbix and MariaDB (with the `Innodb` option) and import the configurations. Zabbix has an import/export feature with `XML` files for `hosts`, `templates`, `valuemaps`, `maps` and `screens`, but the other configurations would have to be done by hand. It was decided that the second option would be the best, mainly because of the fresh install advantages, such as installing a newer Zabbix version and a better knowledge in how to properly setup the system. Enabling `InnoDB` is done by stopping the `mariadb` service, editing the `my.cnf` file and entering the following command:

```
# joe /etc/my.cnf
# Enable the File-Per-Table tablespaces.
innodb_file_per_table=1
```

The export and import features can be seen in Figures 4.4 and 4.5.

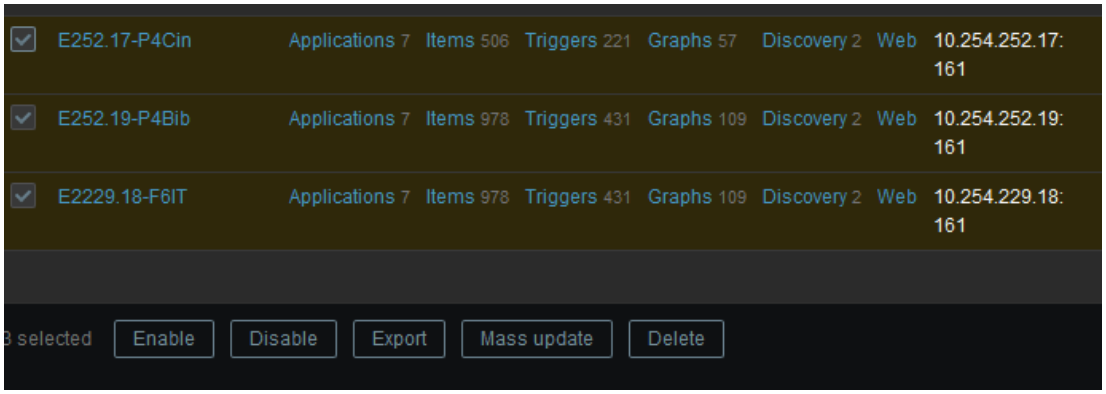


Figure 4.4 – Zabbix export data feature.

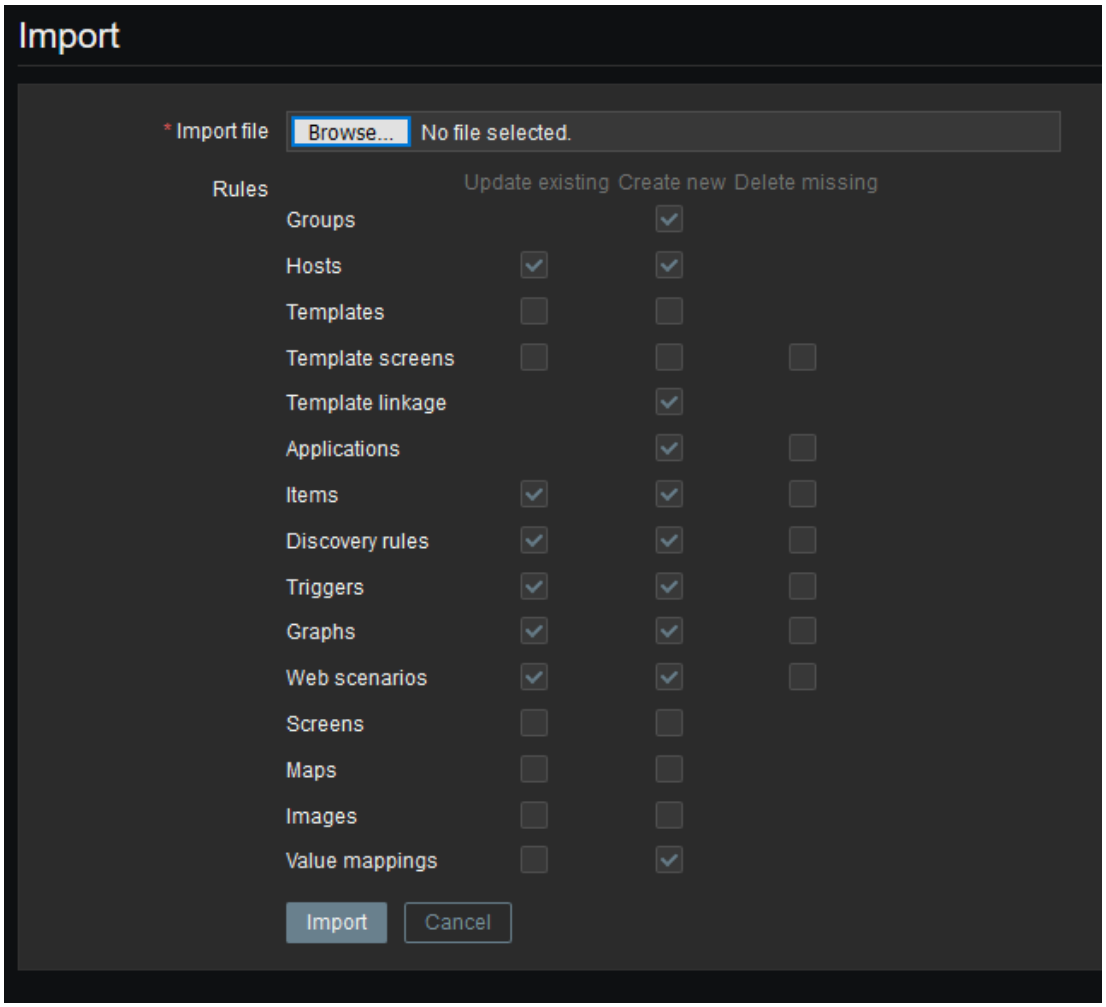


Figure 4.5 –Zabbix import data feature.

Once the fresh install was finished and all the data imported from the `XML` files, Zabbix started to produce successful monitoring results, but another issue still had to be addressed, a performance issue. Zabbix was very slow in producing monitoring outputs, especially in the latest data and graphs section if more than one device was involved. It would sometimes take more than a minute to produce a graph and even more if the date limits were changed. This problem was solved by installing a SSD hard drive and moving the database to a different location. There would be no need to reinstall Zabbix and the database, but simply mount the new hard drive and change a few settings on the configuration files. The SSD was added to VirtualBox and mounted on the new `/mysql` directory. Next the `zabbix-server` and `mariadb` processes where shut down and the path to the new directory was changed in the following configuration files:

```
# joe /etc/my.cnf
[mysqld]
socket=/mysql/mysql.sock
[client]
socket=/mysql/mysql.sock

# joe /etc/php.ini
pdo_mysql.default_socket=/mysql/mysql.sock
mysql.default_socket = /mysql/mysql.sock
mysqli.default_socket = /mysql/mysql.sock

# joe zabbix_server.conf
DBSocket=/mysql/mysql.sock
```

Lastly, all the files in initial `mysql` directory were moved to the new location. After restarting the `zabbix-server` and `mariadb` processes and it was immediately verified that Zabbix was responding much more rapidly. The `Latest data` web page was firing almost instantly, and the graphs were produced much faster.

4.3. Monitoring data

The main monitoring page in the Zabbix frontend is the dashboard. The dashboard is designed to display summaries of all the important information. A dashboard consists of widgets and each widget is designed to display information of a certain kind and source, which can be a summary, a map, a graph, the clock, *etc*. [34]. The widgets can be added or edited, so a single dashboard can contain widgets from various sources, allowing for a quick overview of diverse information. There is also the possibility to create various different dashboards, accessible through the `All dashboards` link under `Global View`. Figure 4.6 illustrates the `Global View` dashboard.

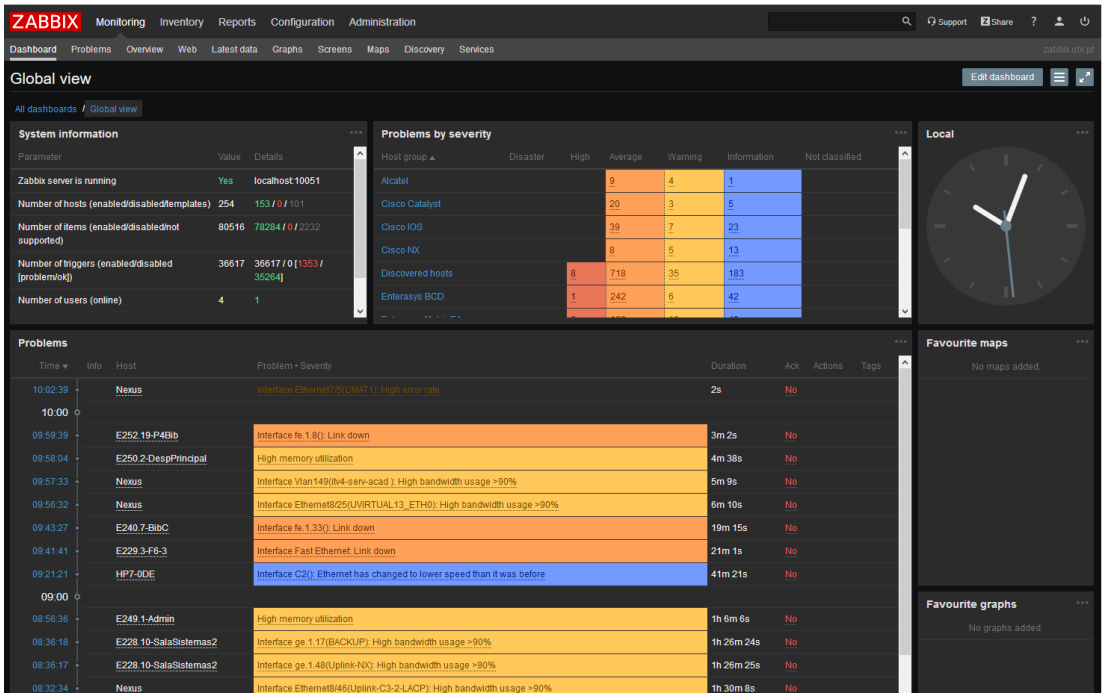


Figure 4.6 - Zabbix main dashboard page.

The `Problems by severity` widget compiles the issues by importance; assign a color to differentiate the severity, so the first step was to analyze the issues marked as High. By hovering the mouse over the number of issues (shown in the Problems by severity table in Figure 4.6), the user can see the offending issues in more detail as shown in the Figure 4.7.

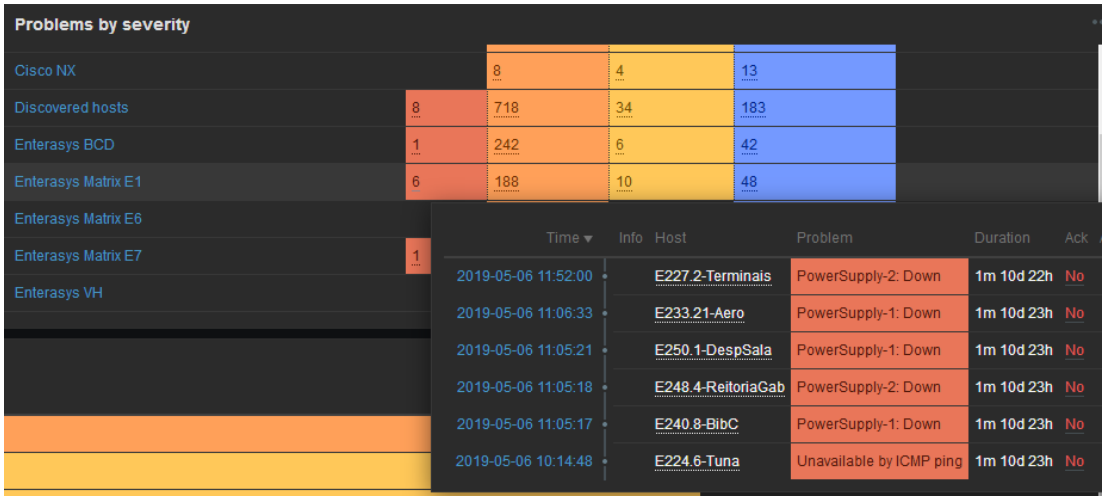


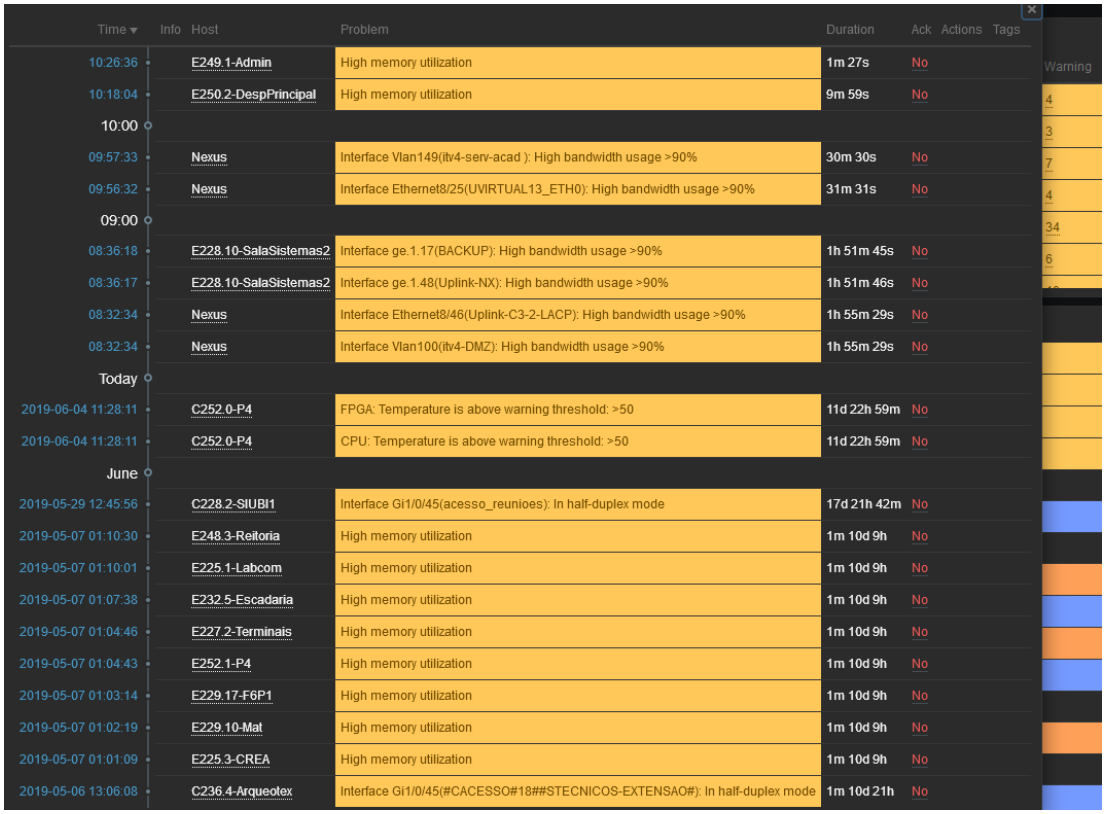
Figure 4.7 – Zabbix high severity alerts.

The details information showed five Enterasys E1 and one E6 switch, which had a faulty PSU, jeopardizing their redundancy. This problem would have to be immediately addressed because if the second PSU goes down, then a network outage will occur in the area of coverage of that device. This allowed the team to implement a pro-active measure as it addressed an issue before it became a problem.

The issues marked `Average` in the color orange were the most frequent and practically all due to link status change, but once again, these issues would be filtered by future configurations described in the Conclusions chapter. This situation is shown in Figure 4.8.

The severities marked Warning in yellow color had some pressing issues, most importantly, High memory utilization and High bandwidth usage as shown in the Figure 4.9.

Figure 4.8 – Zabbix average severity alerts



Time	Info	Host	Problem	Duration	Ack	Actions	Tags
10:26:36		E249.1-Admin	High memory utilization	1m 27s	No		
10:18:04		E250.2-DespPrincipal	High memory utilization	9m 59s	No		
10:00							
09:57:33		Nexus	Interface Vlan149(itv4-serv-acad): High bandwidth usage >90%	30m 30s	No		
09:56:32		Nexus	Interface Ethernet8/25(UVIRTUAL13_ETH0): High bandwidth usage >90%	31m 31s	No		
09:00							
08:36:18		E228.10-SalaSistemas2	Interface ge.1.17(BACKUP): High bandwidth usage >90%	1h 51m 45s	No		
08:36:17		E228.10-SalaSistemas2	Interface ge.1.48(Uplink-NX): High bandwidth usage >90%	1h 51m 46s	No		
08:32:34		Nexus	Interface Ethernet8/46(Uplink-C3-2-LACP): High bandwidth usage >90%	1h 55m 29s	No		
08:32:34		Nexus	Interface Vlan100(itv4-DMZ): High bandwidth usage >90%	1h 55m 29s	No		
Today							
2019-06-04 11:28:11		C252.0-P4	FPGA: Temperature is above warning threshold: >50	11d 22h 59m	No		
2019-06-04 11:28:11		C252.0-P4	CPU: Temperature is above warning threshold: >50	11d 22h 59m	No		
June							
2019-05-29 12:45:56		C228.2-SIUBI1	Interface Gi1/0/45(acesso_reunioes): In half-duplex mode	17d 21h 42m	No		
2019-05-07 01:10:30		E248.3-Reitoria	High memory utilization	1m 10d 9h	No		
2019-05-07 01:10:01		E225.1-Labcom	High memory utilization	1m 10d 9h	No		
2019-05-07 01:07:38		E232.5-Escadaria	High memory utilization	1m 10d 9h	No		
2019-05-07 01:04:46		E227.2-Terminais	High memory utilization	1m 10d 9h	No		
2019-05-07 01:04:43		E252.1-P4	High memory utilization	1m 10d 9h	No		
2019-05-07 01:03:14		E229.17-F6P1	High memory utilization	1m 10d 9h	No		
2019-05-07 01:02:19		E229.10-Mat	High memory utilization	1m 10d 9h	No		
2019-05-07 01:01:09		E225.3-CREA	High memory utilization	1m 10d 9h	No		
2019-05-06 13:06:08		C236.4-Arqueotex	Interface Gi1/0/45(#CACESSO#18##STECNICOS-EXTENSAO#): In half-duplex mode	1m 10d 21h	No		

Warning 4 3 7 4 34 6

Figure 4.9 - Zabbix warning severity alerts

The `High memory` warning could be the normal functioning of the switch, but further investigation would be necessary to confirm this. The `High bandwith usage` warning is concerning and should also be investigated. It could be only a one-off situation due to a system update or file transfer, but in a worst-case scenario, it can also be a broadcast storm or an attack. The best way to investigate this situation is by using a Zabbix graph to map out the traffic in a timeline and see the duration.

The viewing of maps can be done through the `Monitoring>Graphs` menu and by filtering the desired device interface. According to the severity issues widget, the Nexus Eth8/46 interface had high bandwidth usage, so this interface was investigated as shown Figure 4.10.

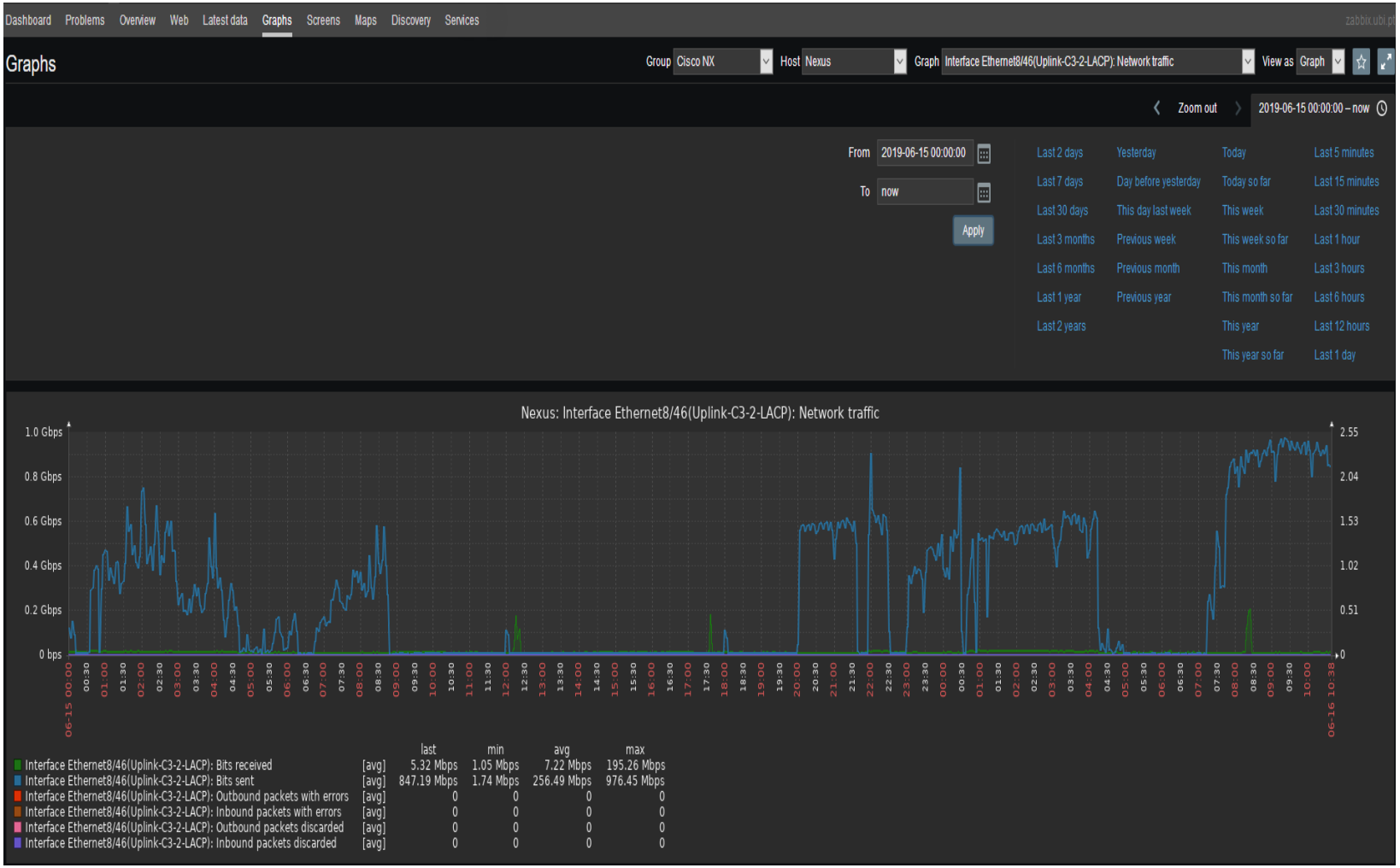


Figure 4.10 – Zabbix interface traffic graph.

The graph was programed with a 2-day interval to see if there was a trend in the traffic. It was seen spikes or bursts of traffic lasting a couple of hours in the bits sent traffic (blue chart line in Figure 4.10). The interface description indicated that this interface was part of a 2 Gbps LACP port-channel group of interfaces connected to a switch that connects all of the main servers, so it was expected to have a lot of traffic. Further inspection could be accomplished by analyzing the server switch ports and speaking with the server administration colleague. This graph could be added to the Zabbix favorite's widget group to more easily be monitored daily. If necessary, the LACP group could be expanded to 3 Gbps.

Another interesting feature of Zabbix is the Screens menu, where global screens and slide shows can be configured. A `Distribution links` screen was configured to monitor the main 10 Gbps distribution links that made up the backbone on the university's network. Figures 4.11, 41.2 and 4.13 show this screen.

An interesting issue was found by analyzing the backbone traffic. There seemed to be constant traffic incoming to the network core at every hour of the day. The Medical faculty link graph (Uplink-Medicina.10G) presented this most clearly. After some research, it was concluded that this traffic was originating from the IP security cameras spread throughout the campuses.

The Latest data menu is an important feature of the web interface because it can list all of the device's latest monitoring data in a list, organized by application. The user can quickly view the information of one or more device's SNMP data, but the more devices are added to the list the heavier it becomes and slower it takes to generate. A graph can also be easily generated from this list. Figure 4.14 show the latest data page.

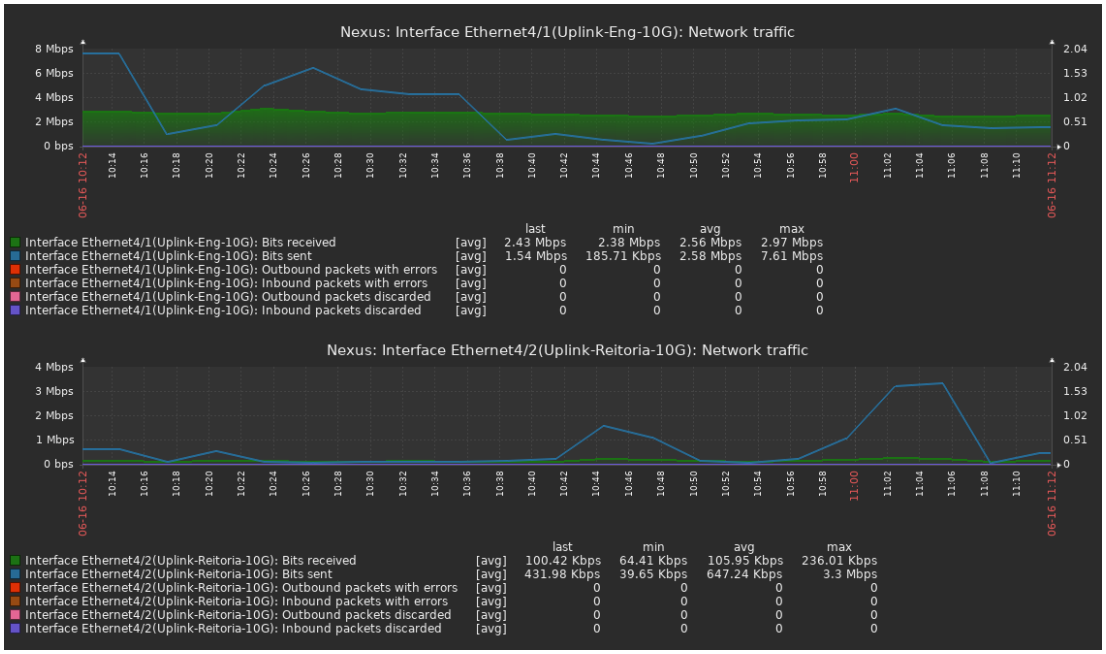


Figure 4.11 – Zabbix distribution links screen.

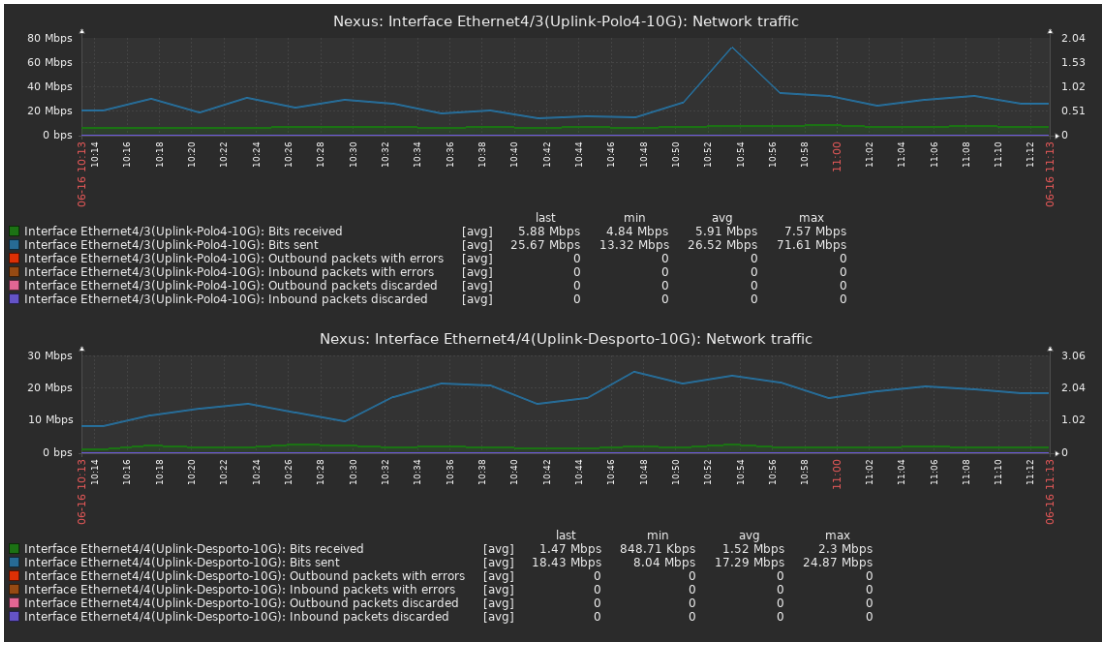


Figure 4.12 - Zabbix distribution links screen.

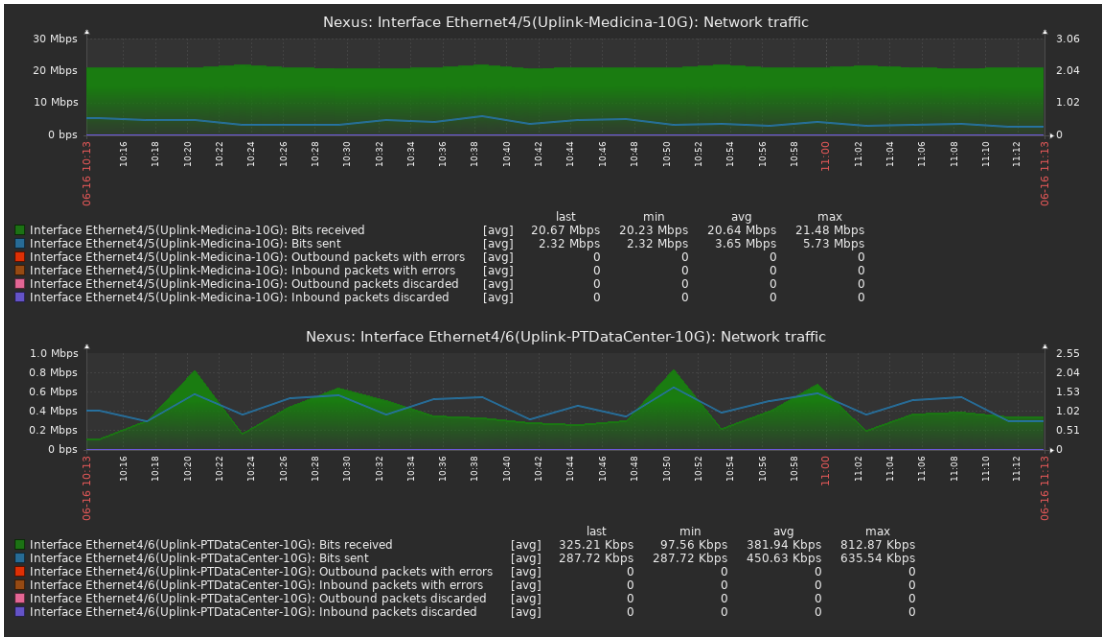


Figure 4.13 - Zabbix distribution links screen.

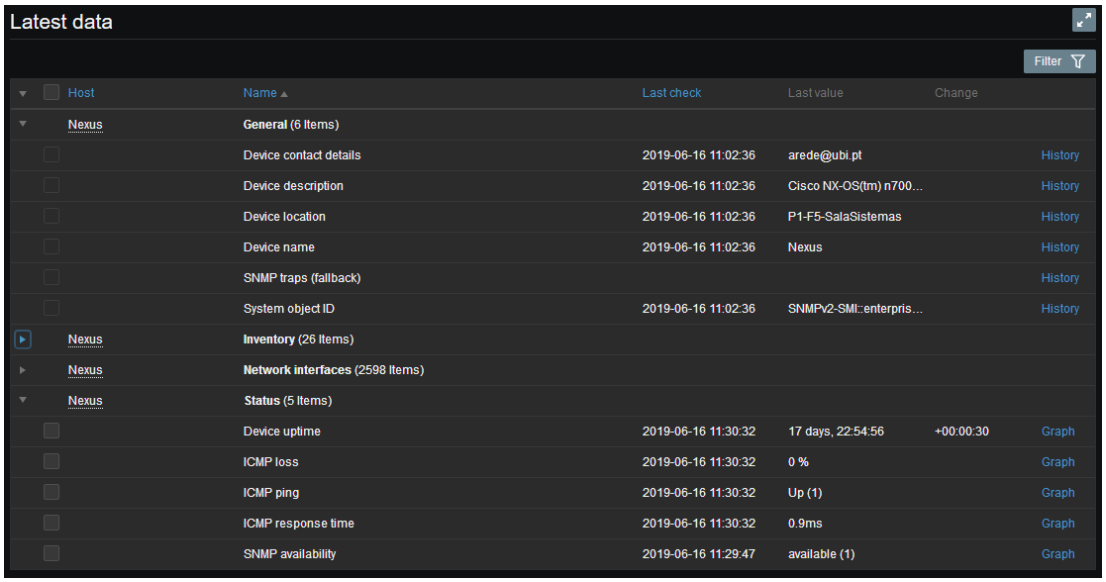


Figure 4.14 – Zabbix latest data web page.

Two more statistics are interesting to analyze. The first one is the system information widget, shown in the initial dashboard. It contains an overview of the number of monitored data, such as hosts, items and triggers. Zabbix was at the moment monitoring 153 devices as shown in Figure 4.15.

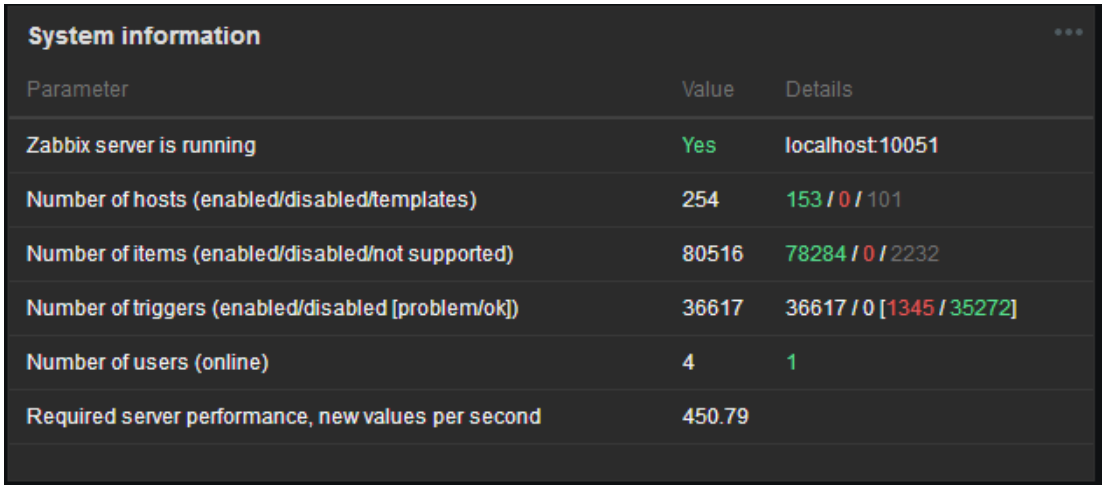


Figure 4.15 – Zabbix system information widget.

The last statistic is the Zabbix server health dashboard accessible through the `All dashboards` link. A few warnings were showing up in regard to processes occupying more than 75% of memory cache. It was interesting to see the number of values processed per second reaching over 600. Further investigation would have to be made to understand the normal accepted values. Increasing the template polling time could help in lowering these values. Figure 4.16 illustrates these statistics.

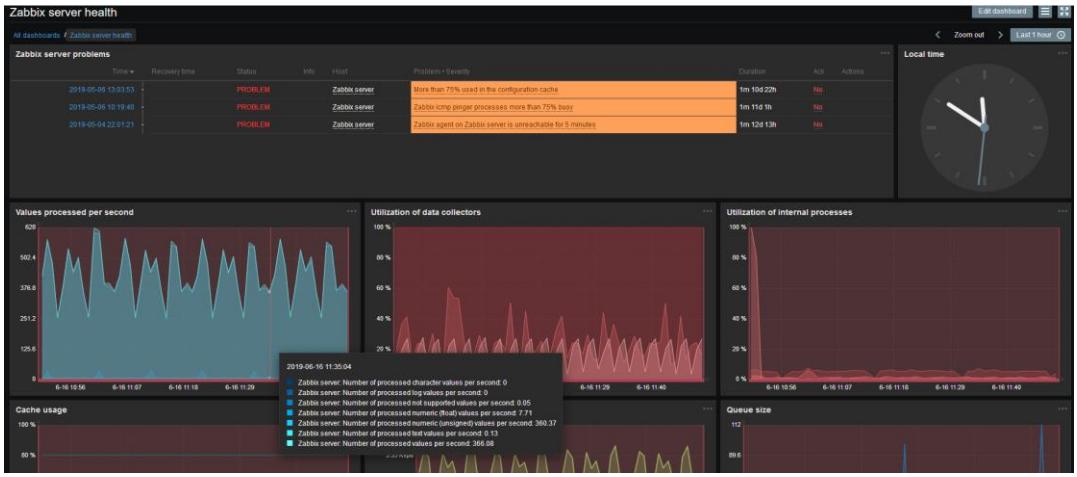


Figure 4.16 – Zabbix server health information dashboard.

To finalize this chapter, it is necessary to point out another advantage to this system regarding device inventory. The Inventory template indexes all of the network device's information regarding models, operating system

version, and serial numbers. This is an important feature because now there is an update to date inventory of equipment and a history of firmware updates. The inventory is accessible through the `Inventory>Hosts` menu and is illustrated in Figure 4.17.

Host ▲	Group	Name	Type	OS	Serial number A
C1.1-Redes	Cisco IOS, Discovered hosts	C1.1-Redes.redes.ubi.pt		15.0(2)EX5	
C224.2-Academicos	Cisco IOS, Discovered hosts	C224.2-Academicos.redes.ubi.pt		15.2(2)E6	
C227.11-FisSec	Cisco IOS, Discovered hosts	C227.11-FisSec.redes.ubi.pt		15.2(2)E6	
C227.16-FisicaP4	Cisco IOS, Discovered hosts	C227.16-FisicaP4.redes.ubi.pt		15.2(2)E6	
C227.17-Quimica	Cisco IOS, Discovered hosts	C227.17-Quimica.redes.ubi.pt		15.2(2)E6	
C228.2-SIUBI1	Cisco IOS, Discovered hosts	C228.2-SIUBI1.redes.ubi.pt		15.0(2)EX5	
C228.3-SIUBI2	Cisco IOS, Discovered hosts	C228.3-SIUBI2.redes.ubi.pt		15.0(2)EX5	
C228.253-AMA	Cisco IOS, Discovered hosts	C228.253-AMA.redes.ubi.pt		12.4(22)YB2	FCZ1042704Q
C229.7-Matematica	Cisco IOS, Discovered hosts	C229.7-Matematica.redes.ubi.pt		15.2(2)E6	
C229.9-Informatica	Cisco IOS, Discovered hosts	C229.9-Informatica.redes.ubi.pt		15.2(2)E7	
C232.11-Fablab	Cisco IOS, Discovered hosts	C232.11-Fablab.redes.ubi.pt		15.0(2)EX5	
C233.13-S9.2	Cisco IOS, Discovered hosts	C233.13-S9.2.redes.ubi.pt		15.2(3)E	
C236.4-Arqueotex	Cisco IOS, Discovered hosts	C236.4-Arqueotex		12.2(55)SE5	
C252.4-P4	Cisco IOS, Discovered hosts	C252.4-P4		12.2(55)SE3	
C252.10-P4	Cisco IOS, Discovered hosts	C252.10-P4		12.2(55)SE3	
C254.2-Padres	Cisco IOS, Discovered hosts	C254.2-Padres			
C2279.7-Fase9	Cisco IOS, Discovered hosts	C2279.7-Fase9.redes.ubi.pt		15.2(2)E6	

Figure 4.17 – Zabbix device inventory page.

4.4. Conclusion

This chapter has described the evaluation, testing process and obtained results of the Zabbix monitoring solution. Although some performance issues were encountered, Zabbix proved to be a stable and reliable platform. The alerting features also proved to be a valuable asset in assisting the network administrator's work.

Chapter 5

5. Conclusions and future work

5.1. General conclusion

The implementation of the Zabbix network monitoring solution proved to be a valuable asset in assisting the network administrator's work. Proactively monitoring the network will, without a doubt, increase the reliability of the university's network while decreasing the number of performance and outage related issues.

The choice of implementing the Zabbix network monitoring tool turned out to be a wise one, even though some issues had occurred. During the course of this seven month work, two new versions of Zabbix where rolled out, 4.0 and 4.2. This work initially started with version 3.4, but after one month, it was easily updated to version 4.0. The smooth update procedure increased the confidence in the product. After a few months, due to the database issue described in the last chapter, it was decided to do a fresh install with the new 4.2 version.

Almost immediately after importing hosts, critical issues were discovered and solutions started to be prepared. There is still some work to be done, as described in the following section, but also, the creation of more screens and graphs is necessary to further analyze the network.

5.2. Future Work

As discussed in the previous chapter, Zabbix was showing unwanted information, such as interfaces status changing between states or passing to a lower speed. These interfaces are connected to user computers and are not very interesting to monitor. The Zabbix console becomes very populated,

which overshadows the important information and the database grows with unnecessary history data. Therefore, it is necessary to identify important interfaces that are crucial to maintain a good network performance and run important services. These interfaces should be correctly identified as to what they connect to, being uplinks to other switches and routers; or important end devices such as wireless access points, printers, servers, IP phones, access controls and IP cameras. If these interfaces could be correctly identified with a naming convention then by configuring the Zabbix templates, a filtering function could be implemented to exclude all the lesser important interfaces.

To implement this feature the following steps will have to be followed:

- Every interface not in used should be set in default setting and administratively shut down
- Create a naming convention for the interfaces to be monitored
- Apply a regex function in the Zabbix filtering option in the Network Interfaces Discovery template.

The naming convention should be applied to the network device's interface description, identifying the type of connection (uplink or end device) and in case of an uplink, identify the remote device port. The identification of the remote port is important because in case of failure, the network administrator can easily identify what is on the other end. The regular expressions function should correctly identify this text and consequently the discovery function indexing the interface.

The configuration of traps is another feature that can be implemented in Zabbix. This could help detect some issues that occur in the monitored host between the manager query intervals. Receiving SNMP traps in Zabbix relies on the `snmptrapd` application and on `snmptt`, which is one of the built-in mechanisms for passing the traps to Zabbix.

Another aspect which can be improved is the performance issue. The Zabbix architecture can be switched to a more complete one by dividing the services (Zabbix, database, HTTP) in different servers. The introduction of Zabbix proxies can help to remove the workload from the Zabbix server. The Zabbix documentation also states that if the monitored host are over 1000 then the operating system should be RedHat Enterprise.

References

[1] "Basics of Network Monitoring," [Online]. Available: www.manageengine.com/network-monitoring/basics-of-network-monitoring.html. [Accessed 05 2019].

[2] "RFC 1157," [Online]. Available: https://tools.ietf.org/html/rfc1157. [Accessed 05 2019].

[3] "RFC 1901," [Online]. Available: https://tools.ietf.org/html/rfc1901. [Accessed 05 2019].

[4] "RFC 1908," [Online]. Available: https://tools.ietf.org/html/rfc1908. [Accessed 05 2019].

[5] "RFC 2273," [Online]. Available: https://tools.ietf.org/html/rfc2273. [Accessed 05 2019].

[6] "RFC 2275," [Online]. Available: https://tools.ietf.org/html/rfc2275. [Accessed 05 2019].

[7] "Best Open Source Network Monitoring Tools and Software," [Online]. Available: https://www.ittsystems.com/best-open-source-network-monitoring-tools/. [Accessed 05 2019].

[8] "Top 5 open source network monitoring tools," [Online]. Available: https://opensource.com/article/19/2/network-monitoring-tools. [Accessed 05 2019].

[9] "7 Best Open Source Network Monitoring Tools for Windows and Linux," [Online]. Available: https://techwiser.com/open-source-network-monitoring-tools/. [Accessed 05 2019].

[10] "The Top 5 Free and Open Source Network Monitoring Software," [Online]. Available: https://blog.capterra.com/top-open-source-free-network-monitoring-software/. [Accessed 05 2019].

[11] "Best Open-Source Network Monitoring Software and Tools," [Online]. Available: https://www.pcwdld.com/best-open-source-network-monitoring-software. [Accessed 05 2019].

[12] "What is Zabbix," [Online]. Available: https://www.zabbix.com/features. [Accessed 05 2019].

[13] "NagiosReport," [Online]. Available: https://community.spiceworks.com/products/17706-nagios. [Accessed 05 2019].

[14] "Icinga Monitoring," [Online]. Available: https://icinga.com/products/. [Accessed 05 2019].

[15] "LibreNMS features," [Online]. Available: https://www.librenms.org/. [Accessed 05 2019].

[16] "Nagios vs. Zabbix," [Online]. Available: https://www.comparitech.com/net-admin/nagios-vs-zabbix/. [Accessed 05 2019].

[17] "Switching from Icinga to Zabbix," [Online]. Available: https://kartaca.com/en/why-we-switched-from-icinga-to-zabbix/. [Accessed 05 2019].

[18] "Zabbix Installation Requirements," [Online]. Available: www.zabbix.com/documentation/4.2/manual/installation/requirements. [Accessed 01 04 2019].

[19] "The CentOS Project," [Online]. Available: https://www.centos.org/. [Accessed 05 2019].

[20] "Rufus," [Online]. Available: https://rufus.ie/. [Accessed 05 2019].

[21] "CentOS download," [Online]. Available: www.centos.org/download/.

[22] "VirtualBox," [Online]. Available: https://www.virtualbox.org/. [Accessed 05 2019].

[23] "CentOS Install," [Online]. Available: www.tecmint.com/things-to-do-after-minimal-rhel-centos-7-installation/. [Accessed 05 04 2019].

[24] "Zabbix Installation," [Online]. Available: www.zabbix.com/documentation/4.2/manual/installation. [Accessed 05 04 2019].

[25] "How to Install Zabbix," [Online]. Available: https://computingforgeeks.com/how-to-install-zabbix-server-4-0-on-centos-7/. [Accessed 05 04 2019].

[26] "Installing Zabbix," [Online]. Available: https://omgfoss.com/install-zabbix-centos-7/. [Accessed 05 04 2019].

[27] "Installing Zabbix," [Online]. Available: https://www.howtoforge.com/tutorial/centos-zabbix-system-monitoring/. [Accessed 05 04 2019].

[28] "Configuring Host Groups," [Online]. Available: www.zabbix.com/documentation/4.0/manual/config/hosts. [Accessed 04 2019].

[29] "Configuring Templates," [Online]. Available: www.zabbix.com/documentation/4.0/manual/config/templates. [Accessed 04 2019].

[30] "Configuring Triggers," [Online]. Available: www.zabbix.com/documentation/4.0/manual/config/triggers. [Accessed 04 2019].

[31] "Nesting Templates," [Online]. Available: www.zabbix.com/documentation/4.2/manual/config/templates/nesting. [Accessed 05 2019].

[32] "iReasoning Software," [Online]. Available: http://www.ireasoning.com/. [Accessed 05 2019].

[33] "Configuring Graphs," [Online]. Available: www.zabbix.com/documentation/4.2/manual/config/visualisation/graphs. [Accessed 05 2019].

[34] "Zabbix Dashboard," [Online]. Available: www.zabbix.com/documentation/4.2/manual/web_interface/frontend_sections/monitoring/dashboard. [Accessed 05 2019].

[35] "Rufus software," [Online]. Available: rufus.ie.