УДК 004.415

П.Д. Макаров, 25.makarovv@gmail.com
С.В. Зотов, sergeyzotoff11@gmail.com
Кыргызско-Германский институт прикладной информатики

АВТОМАТИЧЕСКОЕ ДОБАВЛЕНИЕ И СОРТИРОВКА ХОСТОВ В ZABBIX

В данной статье исследуется задача автоматизации процесса добавления новых узлов сети (хостов) в систему мониторинга Zabbix и их автоматической сортировки по группам. Выделяются три основные проблемы при использовании стандартных средств Zabbix: ограниченная классификация хостов, сложность фильтрации лишних данных и высокая нагрузка на сервер при сканировании большой адресной области. Проведен анализ существующих возможностей Zabbix (автообнаружение сети и авторегистрация агентов) и показаны их ограничения. В качестве решения предлагается внешний скрипт, интегрированный с Zabbix через API, который автоматически добавляет устройства в мониторинг и распределяет их по заранее определенным группам. Предложенный подход позволяет учесть тип устройства и его расположение при добавлении, а также заполнить дополнительные инвентарные данные. В работе описана архитектура Zabbix и используемые протоколы (Zabbix Agent, SNMP, ICMP), подробно изложен алгоритм предлагаемого решения и приведено его сравнение с существующими аналогами. Проведены экспериментальные испытания на тестовом стенде; результаты демонстрируют успешное добавление и группировку хостов без участия администратора. Представлены графики распределения хостов по группам, снимки экрана работы скрипта и интерфейса Zabbix. Решение обеспечивает более гибкое и масштабируемое добавление хостов в мониторинг по сравнению со стандартными методами, что подтверждается результатами эксперимента.

Ключевые слова: Zabbix, автообнаружение, авторегистрация, MAC-адрес, ARP-запросы, OUI, Zabbix API, мониторинг, классификация.

Введение

Современные предприятия опираются на IT-инфраструктуру, включающую десятки и сотни серверов, рабочих станций и сетевых устройств, требующих постоянного мониторинга. Системы мониторинга, такие как Zabbix, позволяют автоматически собирать метрики производительности серверов, сетевого оборудования и сервисов. Zabbix является открытым и масштабируемым решением для наблюдения за разнообразными элементами ИТ-инфраструктуры – от сетевых интерфейсов до виртуальных машин. Использование таких систем значительно повышает оперативность обнаружения проблем и снижает простой сервисов [1, 2]. В условиях усложнения ИТ-инфраструктуры и роста объёма событий возрастает потребность в централизованном сборе и обработке данных мониторинга, а также в наглядной визуализации для оперативного реагирования [14].

Однако начальная настройка мониторинга крупных сетей сопряжена с существенными трудозатратами. Администратору приходится вручную добавлять каждый хост (сервер, маршрутизатор и пр.) и назначать ему правильные группы и шаблоны мониторинга. При увеличении числа узлов этот процесс становится долгим и подверженным ошибкам. Задача состоит в том, чтобы автоматизировать добавление новых устройств в мониторинг и создать инструмент, который решил бы проблему их правильной классификации без участия человека.

В существующих версиях Zabbix предусмотрены механизмы автообнаружения и авторегистрации, облегчающие ввод хостов в систему. Однако при использовании этих стандартных функций выделяется три проблемы: во-первых, не всегда удается автоматически определить тип устройства и назначить ему нужный шаблон (ограниченные возможности классификации); во-вторых, существуют сложности с фильтрацией лишних «ложных» хостов (например, виртуальных интерфейсов), которые могут добавляться при сетевом сканировании; в-третьих, массовое сканирование широких диапазонов IP-адресов создает значительную нагрузку на сервер мониторинга. В данной работе приведены сложности более подробно и предложено решение, устраняющее указанные недостатки [3, 6].

Формулировка проблемы

В стандартной поставке Zabbix имеются два основных механизма для автоматического добавления хостов: автообнаружение сети (Network Discovery) и авторегистрация агентов (Active Agent Auto-registration). Оба подхода снижают ручной труд, однако не лишены недостатков [3, 6].

При использовании автообнаружения сети выделяется три проблемы:

1. Избыточные хосты при сканировании. Zabbix-сервер активно сканирует указанный диапазон IP-адресов и добавляет все, что отвечает на запросы. В результате могут автоматически добавляться не только сами устройства, но и их виртуальные интерфейсы. Например, в случае сетевого оборудования Cisco, при опросе SNMP без дополнительной фильтрации, Zabbix способен воспринимать каждый VLAN-интерфейс как отдельный хост. Это ведет к появлению лишних записей и затрудняет администрирование.
2. Высокая нагрузка при широком диапазоне. Автообнаружение методом сканирования плохо масштабируется на большие сети. Если диапазон адресов велик (например, вся подсеть 10.0.0.0/8 с миллионами адресов), то непрерывное сканирование потребляет очень много времени и ресурсов сервера. Существуют сложности с применением штатного дискавери на крупных распределенных сетях – в реальном кейсе сеть из 14000 узлов потребовала отказаться от штатного автообнаружения из-за неприемлемой нагрузки [5].
3. Недостаток информации о хосте. Механизм discovery добавляет хосты с минимальным набором данных – по сути, только IP-адрес (и, возможно, SNMP OID для определения типа устройства). Он не позволяет автоматически заполнить «нетехническую» информацию: имя или описание объекта, местоположение, ответственных лиц и т.д. Для эффективного мониторинга часто требуется классифицировать узлы не только по техническим параметрам, но и по бизнес-признакам (офис, филиал, критичность). Стандартное автообнаружение не решает задачи добавления подобных атрибутов [3].

Схожие существуют сложности и при авторегистрации активных агентов. Данный механизм срабатывает, когда на сервер подключается неизвестный Zabbix-агент. Можно настроить действие авторегистрации, чтобы новый хост автоматически добавился в группу и получил шаблон. Но задача состоит в том, что решение годится только для серверов и ПК с установленным агентом. Оно не охватывает оборудование, работающее по SNMP (коммутаторы, принтеры и пр.) или не поддерживающее агент. Кроме того, инструмент, который встроен в Zabbix для авторегистрации, обладает ограниченными критериями: он может условно сортировать хосты по полученным от агента данным (имя хоста, метка, IP-адрес), но, например, не узнает автоматически, что за тип системы подключился – веб-сервер, база данных или маршрутизатор [6].

Таким образом, имеющиеся средства автоматизации в Zabbix покрывают не все случаи. Требуется инструмент, который решил бы сразу обе задачи: и обнаружение сетевых устройств без агентов, и глубокую классификацию добавляемых хостов по типам и группам.

Анализ существующих решений

Для решения проблемы автоматического введения множества узлов в мониторинг можно выделить несколько подходов:

- Ручное периодическое добавление. На практике до сих пор нередко используется полуавтоматический подход: администратор по расписанию сканирует сеть сторонними утилитами (например, nmap), выявляет новые устройства и вручную добавляет их в Zabbix. Этот метод крайне трудоемкий и неоперативный, поэтому в современных условиях не рассматривается как удовлетворительный.
- Встроенное автообнаружение Zabbix. Как отмечалось, Network Discovery позволяет настроить правила сканирования: например, пинг (ICMP) для выявления активных

IP и запросы по SNMP для опознания типа устройства. Далее через Actions можно задать, как обрабатывать найденные узлы – добавить в определенную группу, присвоить шаблон и т.д. Данный способ хорошо работает на небольших диапазонах и простых сетях. Его минусы – трудности масштабирования и риск «мусорных» хостов (без фильтра, как обсуждалось, система может добавить лишние узлы, например интерфейсы). Также требуются ресурсы на регулярный обход сети, что отмечено в одном из ресурсов: «The downside of auto-discovery is the time and resources need to constantly scan» [6].

- Авторегистрация агентов. Это встроенное решение для серверов и рабочих станций с Zabbix Agent в активном режиме. Когда новый агент запущен и знает адрес Zabbix-сервера, он сам отправляет запрос на регистрацию. Сервер, получив от неизвестного агента его hostname и метаданные, выполняет действие авторегистрации: помещает хост в заданную группу, привязывает шаблоны, отправляет уведомление администратору и т.п. Преимуществом является отсутствие сканирования – система реагирует только на реальные подключения, что разгружает сервер. Однако авторегистрация не может определить категорию устройства за пределами той информации, что передал агент. Если метаданные не настроены должным образом, все новые хосты окажутся в одной группе по умолчанию, и администратору придется вручную рассортировывать их позже. В сообществе отмечают, что автодействия при регистрации ограничены: можно лишь добавить в группу и привязать шаблон, тогда как discovery-правила дают больше возможностей обработки [6].
- Сторонние скрипты и интеграции. Многие администраторы создают собственные скрипты, используя API Zabbix или CMDB-системы. Например, встречаются решения на Ansible или Python, выгружающие списки хостов из внешней базы (инвентаризации) и массово загружающие их в Zabbix через API. Подход, реализованный командой X5 Retail Group, описан в открытых источниках: данные об объектах брались из корпоративной системы (SAP), преобразовывались в XML, и через вызовы Zabbix API хосты создавались автоматически [5]. Такие скрипты гибки – они позволяют добавить любую необходимую информацию (например, привязать адрес, ID филиала, контактное лицо к полям инвентаря в Zabbix). Однако разработка и сопровождение собственного скрипта требует определенной квалификации в программировании.

Сравнение подходов. В таблице 1 обобщены ключевые характеристики встроенных методов Zabbix и предлагаемого в данной работе решения.

Таблица 1– Сравнение методов автоматического добавления хостов в Zabbix

Критерий	Автообнаружение (Network Discovery)	Авторегистрация (Agent Autoregistration)	Предлагаемое решение (скрипт)
Способ обнаружения устройств	Активное сканирование сети (ICMP, SNMP)	Пассивное: хост сам подключается (агент)	Целевое сканирование и/или загрузка списка узлов из внешнего источника
Требуемое ПО на хосте	Не требуется (SNMP-агент обычно встроен)	Требуется Zabbix Agent на хосте	Не требуется для SNMP-устройств; для серверов может использовать агент
Гибкость классификации при добавлении	Ограниченная: по IP или SNMP OID, простые правила	Ограниченная: по имени хоста или метаданным от агента	Высокая: настраиваемые критерии в коде (тип устройства,

			местоположение и др.)
Нагрузка на сервер	Значительная при больших диапазонах	Низкая (сканирование не используется)	Умеренная: скрипт опрашивает только известные узлы или ограниченный диапазон
Дополнительные данные (инвентарь, описания)	Минимальные (IP, базовый SNMP-идентификатор)	Минимальные (имя хоста, IP, метка агента)	Расширяемые: заполняются при добавлении (имя объекта, адрес, отдел и пр.)
Масштабируемость	Плохо масштабируется на тысячи узлов	Хорошо масштабируется (добавление по мере подключения агентов)	Хорошо масштабируется при наличии актуального списка узлов; сканирование распределяется во времени

Как видно из таблицы 1, ни один из штатных механизмов не удовлетворяет полностью всем требованиям. В качестве решения предлагается реализовать интегрированный подход: использовать скрипт, который комбинирует преимущества обоих методов (поддержка SNMP и агентов) и добавляет недостающую гибкость. Перед реализацией такого решения необходимо понять, как Zabbix взаимодействует с хостами и какими техническими средствами можно воспользоваться.

Zabbix и используемые протоколы

Система мониторинга Zabbix имеет классическую серверно-агентскую архитектуру. Zabbix-server – центральный компонент, который собирает данные, вычисляет триггеры (события) и осуществляет оповещения. Данные конфигурации и истории хранятся в базе данных (СУБД, например MySQL или PostgreSQL). Для взаимодействия пользователя с системой служит веб-интерфейс Zabbix (Frontend), обычно размещаемый на том же сервере. Также может использоваться промежуточный слой – Zabbix-proxy, который собирает данные в удаленных сегментах сети и передает их на главный сервер, разгружая последний [2, 3].

Zabbix-agent – это легкое программное приложение (демон), устанавливаемое на мониторируемом хосте (например, на сервере под управлением Windows или Linux). Агент собирает локальные метрики: загрузку CPU, использование памяти, состояние дисков, работу приложений и т.д. [7] В пассивном режиме агент ожидает запросов от сервера, а в активном – сам периодически отправляет данные на сервер. Соединение агента с сервером или прокси происходит по протоколу TCP (порт 10050 для пассивного агента, 10051 для активного). Обмен данными идет в проприетарном формате Zabbix, оптимизированном для мониторинга [7, 3].

Помимо собственных агентов, Zabbix умеет опрашивать устройства по стандартным протоколам. Один из основных – SNMP (Simple Network Management Protocol), широко используемый для сетевого оборудования [4]. SNMP относится к прикладному уровню и предоставляет единый язык обмена информацией между устройствами и системой управления. В контексте Zabbix сервер может выступать в роли SNMP-менеджера, который запрашивает у SNMP-агента (запущенного на устройстве, например в прошивке маршрутизатора) необходимые параметры. Идентификация показателей происходит через OID (объектные идентификаторы) из MIB-баз. Zabbix поддерживает SNMP v1/2c и v3, позволяя собирать, например, загрузку процессора коммутатора, температуру датчиков,

состояние портов и т.д. Если устройство отвечает на SNMP-запросы, Zabbix рассматривает его как доступный хост даже без установки своего агента [3, 11].

Также Zabbix применяет ICMP (эхо-запросы, «ping») для простейшей проверки доступности хоста. Simple checks (простые проверки) включают в том числе ping – если устройство откликается, фиксируется время отклика и факт его онлайн-статуса. Эти методы не требуют установки ПО на целевом узле [3].

При непосредственном сканировании сети на канальном уровне может участвовать ARP (Address Resolution Protocol). Данный протокол предназначен для преобразования логического IP-адреса в физический MAC-адрес в рамках локальной сети [9]. Он необходим, потому что для отправки данных по локальной сети устройство должно знать MAC-адрес получателя, а не только его IP-адрес. Утилита arp-scan отправляет ARP-запросы на все адреса заданной подсети и выводит список устройств с их IP и MAC-адресами. Каждое сетевое устройство имеет уникальный 48-битный MAC-адрес (Media Access Control). MAC-адрес записывается как шесть шестнадцатеричных байт (например, 00:1A:2B:3C:4D:5E). Первые 24 бита такого адреса – это OUI (Organizational Unique Identifier), уникальный идентификатор организации-производителя, который присваивается регистрирующей администрацией IEEE [8, 10]. Оставшиеся 24 бита определяют конкретное устройство производителя. Все это дает возможность автоматически классифицировать устройства по типам (серверы, маршрутизаторы, коммутаторы и т.п.) и применять соответствующие шаблоны мониторинга.

Для автоматизации управления хостами Zabbix предоставляет программный интерфейс – Zabbix API. Он доступен по HTTP(S) в виде веб-сервиса (обычно http://<server>/zabbix/api_jsonrpc.php) и принимает или возвращает данные в формате JSON-RPC. С помощью API можно удаленно выполнять практически все действия, доступные через веб-интерфейс: создавать, обновлять и удалять узлы, группы, шаблоны, получать метрики и т.д. Например, существует метод host.create, позволяющий программно добавить новый хост с указанием его параметров (имя, IP-адрес, группы, шаблоны и пр.) [3]. API открывает возможности интеграции: внешний скрипт может сам зарегистрировать устройство в Zabbix, минуя ручное добавление. В контексте нашего решения API – ключевой механизм, через который скрипт будет добавлять и настраивать хосты. Использование API требует учетных данных пользователя Zabbix с правами админа и формирования корректных JSON-запросов (с указанием метода, параметров и токена аутентификации). Библиотеки для работы с Zabbix API существуют на различных языках программирования (Python, Go, Perl и др.), что упрощает разработку интеграций [3].

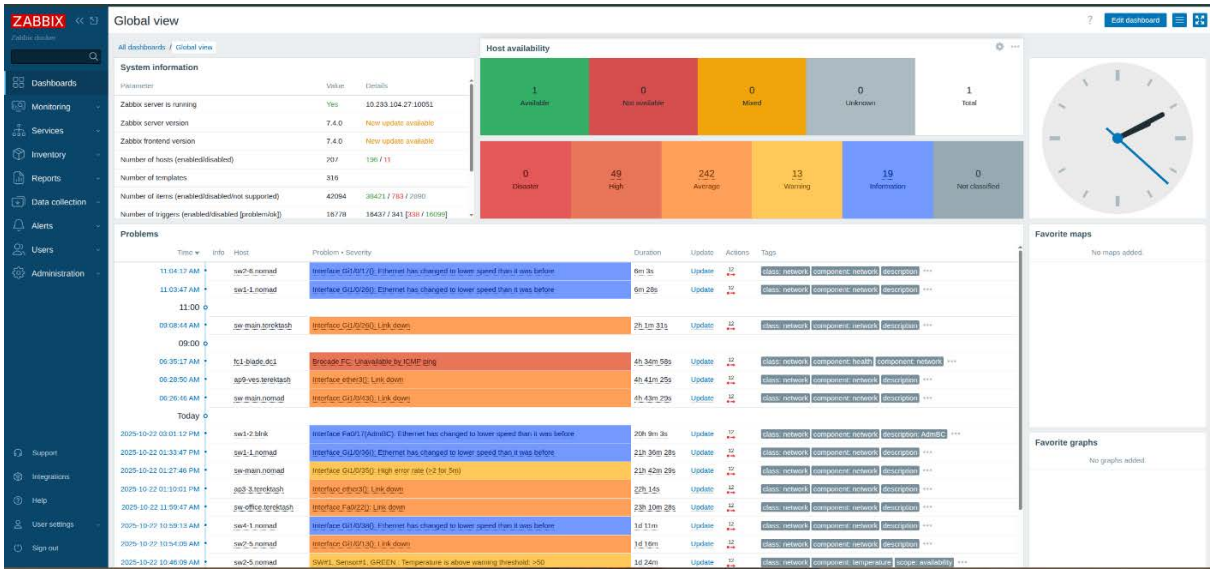


Рисунок 1 – Общая архитектура системы мониторинга Zabbix. Центральный сервер Zabbix управляет сбором данных через агентов (установленных на серверах) и опрашивает

сетевое оборудование по SNMP. При масштабировании используются промежуточные прокси-серверы. Взаимодействие с пользователем осуществляется через веб-интерфейс.

Предлагаемое решение

Исходя из анализа ограничений штатных методов, предлагается доработать процесс автоматического включения узлов в мониторинг путем разработки внешнего сценария (скрипта), взаимодействующего с Zabbix через API. В качестве решения предлагается комбинированный подход: использовать вне Zabbix умный скрипт, который сам обнаруживает новые устройства и классифицирует их по группам перед добавлением в систему. Ниже приведено подробное описание предлагаемого решения и обоснование его эффективности.

1. Источник данных о новых хостах. Скрипт может получать информацию о хостах из нескольких источников:
   - Активный опрос подсети. Для заданных сетевых сегментов скрипт раз в N минут выполняет быстрое сканирование (например, ICMP ping всех адресов). Выявляются новые активные IP, ранее не представленные в Zabbix. Для каждого нового IP может выполняться дополнительный опрос: если на устройстве открыт SNMP, то для определения типа устройства; если откликается Zabbix Agent, то можно собрать имя хоста и метку агента. Такая многоступенчатая проверка позволяет отличить условно маршрутизатор от сервера.
   - Внешний реестр. В случае если у организации есть актуальная база данных устройств (CMDB, IPAM и пр.), скрипт берет список узлов оттуда. Например, это может быть CSV-файл или БД, где указаны IP, имя устройства, тип, местоположение. Тогда скрипт просто сверяет текущий список хостов Zabbix с эталонным списком и находит новые (или отсутствующие) устройства.
   - События авторегистрации. Zabbix способен генерировать внутреннее событие, когда новый агент запросил регистрацию. Наш скрипт может подписываться на такие события (через периодический опрос Zabbix API) и реагировать, выполняя расширенную обработку. Например, при появлении авторегистрации скрипт получит от Zabbix имя хоста и его IP, и далее может сам скорректировать его группы или шаблоны более гибко, чем штатное действие [3].

В реализации на тестовом стенде мы использовали активное сканирование локальной сети и частично данные из файла с перечнем устройств. Это имитирует ситуацию, когда известна структура сети (список оборудования), но требуется автоматическое добавление их в мониторинг.

2. Логика классификации и группировки. После обнаружения нового устройства скрипт определяет его атрибуты:
   - Тип устройства. По результатам SNMP-опроса (например, строка sysDescr содержит слово «Cisco IOS» или OID принадлежит ветви Cisco) скрипт относит узел к категории «сетевое оборудование – Cisco». Если агент передал метку system.uname с указанием ОС Windows или Linux – категория «сервер». Предусмотрены правила для основных типов: коммутаторы, маршрутизаторы, сервера Windows, сервера Linux, принтеры (по SNMP может определяться как Printer MIB) и т.д. [12, 13]
   - Подсеть. IP-адрес анализируется на принадлежность к определённому диапазону, соответствующему филиалу либо сегменту. Например, адреса 10.10.x.x – офис А, 10.20.x.x – офис В. Кроме того, если внешняя база содержит привязку IP к объекту (например, магазин №123), скрипт извлекает этот идентификатор.

- Имя хоста. Скрипт может попытаться получить DNS-имя по IP через обратное разрешение или использовать шаблон именования. В нашем случае, если устройство найдено в CSV-списке, берётся заданное там имя; иначе формируется имя по типу и IP (например, «SW-10.10.1.5» для коммутатора по адресу 10.10.1.5) [3].
- Шаблон мониторинга. Каждой категории устройств соответствует определённый шаблон Zabbix (Template). Для Cisco – шаблон SNMP для Cisco IOS, для Windows – шаблон Windows OS Agent, и т.п. Эти соответствия заданы в конфигурации скрипта [12, 13].

Исходя из этих данных, скрипт готовит структуру нового хоста для добавления: название, IP-интерфейс (тип 1 – агент или 2 – SNMP, в зависимости от типа устройства), принадлежность к группам. Группы в Zabbix заранее созданы администратором, отражая нужную классификацию – например, группы по территориальному признаку («Office A», «Office B») и по типу оборудования («Servers», «Network Devices»). Предлагается доработать существующую структуру групп таким образом, чтобы у каждого добавляемого хоста было как минимум две группы: одна – по географии, вторая – по роли. Скрипт, анализируя признаки, формирует соответствующие записи: например, коммутатор в офисе A попадёт в группы «Network Devices» и «Office A». Если необходимы дополнительные группы (например, «Cisco»), их также можно назначить.

3. Добавление хоста через API. Получив всю информацию, скрипт вызывает метод host.create API Zabbix. В параметрах указывается:
   - «host» – имя узла (string);
   - «interfaces» – список интерфейсов. Для SNMP-устройства добавляется интерфейс типа SNMP с IP и портом 161; для сервера с агентом – интерфейс типа агент с портом 10050. При необходимости можно добавить несколько интерфейсов (например, и агент, и SNMP, если устройство поддерживает оба).
   - «groups» – массив групп, куда включить хост (передаются идентификаторы групп Zabbix).
   - «templates» – массив шаблонов для привязки (идентификаторы шаблонов мониторинга).
   - «inventory_mode» – режим инвентаризации. Включается активный режим, чтобы можно было заполнить поля.
   - «inventory» – словарь полей инвентаря: сюда скрипт может записать доступные данные (имя объекта inventory.name, адрес inventory.location, контакты и др.).

Так формируется запрос JSON-RPC. Zabbix API возвращает либо успешный результат (ID созданного хоста) либо ошибку. Возможные ошибки – хост с таким именем или IP уже существует (на случай гонки или дублирования). Скрипт обрабатывает ошибки: например, при дублировании может выполнить host.update для обновления групп или шаблонов существующего узла [3].

4. Обработка удалений и изменений. В рамках данной задачи основной упор сделан на добавление новых хостов. Однако в промышленной эксплуатации полезно удалять из мониторинга устройства, которые более не активны. При дальнейшей разработке существует возможность добавления в скрипт отметки обнаруженных ранее, но ныне недоступных узлов с последующим уведомлением администратора либо через API помещением их в отдельную группу «Неактивные» или даже удаления спустя время.
5. Запуск и интеграция. Скрипт может работать как фоновый сервис (daemon) или запускаться по расписанию на сервере Zabbix или другом управляемом узле. В экспериментальной реализации использован запуск каждые 5 минут. Также настроены права доступа: для API создан отдельный пользователь с нужными привилегиями на группы – это безопаснее, чем использовать администраторский логин.

Обоснование решения

Предложенный подход решает обозначенные проблемы следующим образом. Во-первых, скрипт отфильтровывает лишние данные: при SNMP-сканировании он учитывает только основные IP-устройства (например, только IP Loopback-интерфейса маршрутизатора, игнорируя VLAN-интерфейсы), что устраняет проблему множественных хостов с одного устройства. Во-вторых, нагрузка на сервер минимизирована – скрипт сканирует адреса небольшими порциями или использует уже известный список, а не перебирает весь диапазон целиком. В нашем стенде 60 адресов проверяются за доли секунды, нагрузка на Zabbix от API-вызовов незаметна (в реальной сети можно масштабировать, запуская несколько потоков или ограничивая частоту добавлений). В-третьих, решение обогащает информацию о каждом хосте: сразу при добавлении заполняются группы, шаблоны и поля инвентаря. Таким образом, новый узел появляется в мониторинге уже в правильном месте с нужным набором проверок. Администратору не требуется выполнять ручную сортировку – Zabbix-dashboard сразу отражает устройство в соответствующей группе (например, новый коммутатор сразу виден в группе «Switches» и его метрики собираются по SNMP-шаблону).

Еще одно достоинство – гибкость и адаптируемость. Правила классификации в скрипте легко изменить под особенности конкретной инфраструктуры. Можно учитывать нетипичные признаки: например, проверять отзывчивость веб-интерфейса устройства (HTTP) для отнесения к определенному шаблону. Также через API можно реализовать привязку триггеров или настройку уведомлений индивидуально под хост, если это требуется. Такой уровень тонкой настройки недоступен штатным механизмам автообнаружения.

Мы использовали открытые библиотеки: pysnmp (для SNMP-запросов) и pyzabbix (для удобства работы с Zabbix API в Python). Это ускорило разработку, так как многие детали (формирование правильного JSON, обработка сессий) берут на себя готовые модули.

Сравнение с существующими аналогами

Рассмотренное решение фактически представляет собой внешнюю надстройку над Zabbix, призванную компенсировать недостатки встроенных средств. Аналоги ему можно встретить в крупных организациях, где стандартный функционал мониторинга дополняют скриптами синхронизации. В частности, описанный опыт X5 Tech близок по идее: там интеграция мониторинга с мастер-данными позволила автоматически добавлять магазины в Zabbix, устранив ручной труд [5]. Наш подход более универсален, так как не зависит от наличия корпоративной БД – он может сам сканировать сеть.

Если сравнивать с другими системами мониторинга, некоторые коммерческие решения (Cisco Prime, SolarWinds Orion) имеют более развитые встроенные функции автообнаружения с классификацией. Однако они зачастую привязаны к конкретным вендорам и все равно требуют ручной настройки правил. Предлагаемое решение превосходит встроенный функционал Zabbix в гибкости, оставаясь при этом достаточно простым: используется скрипт средней сложности, который можно модифицировать под разные требования. В отличие от статических конфигураций, скриптовой подход дает возможность быстро реагировать на изменения – изменить логику группировки, добавить поддержку нового типа устройств и т.д., без ожидания новых версий Zabbix [7].

Таким образом, целесообразно доработать экосистему мониторинга с помощью внешнего инструмента, что выгодно отличает наше решение от стандартных сценариев. По совокупности характеристик (гибкость, масштабируемость, информативность) данное решение является эффективной альтернативой существующим способам автоматического добавления хостов.

Результаты эксперимента

Для экспериментальной проверки разработанного подхода был развернут тестовый стенд. Конфигурация стенда:

- Zabbix Server 6.0 LTS на виртуальной машине (4 CPU, 8 GB RAM, ОС Ubuntu 20.04)

- Zabbix Proxy не использовался (не был необходим при небольшом объеме теста)
- Тестовая сеть из 60 узлов:
  - 10 виртуальных машин Windows с установленным Zabbix Agent, 10 виртуальных машин Linux с агентом, 5 аппаратных маршрутизаторов Cisco, 5 коммутаторов Cisco, 30 устройств Mikrotik (SNMP включен, RouterOS, сетевое оборудование)
  - Адресное пространство – 172.16.224.0/24, всего 256 адресов, из которых активны ~60.
- Скрипт выполнен на Python 3.10, запускается каждые 5 минут на том же сервере, что и Zabbix.

Перед запуском экспериментального сценария в Zabbix вручную были созданы необходимые группы: «Servers», «Network», «Mikrotik», «Cisco», «Windows», «Linux», «Office1», «Office2» и т.д, а также шаблоны: Template OS Windows, Template OS Linux, Template Net Cisco SNMP, Generic ICMP Ping (для устройств, от которых только ping). Новые узлы по замыслу должны автоматически попасть в эти группы и получить соответствующие шаблоны [12, 13].

Ход эксперимента. Изначально Zabbix не содержал ни одного хоста из тестовой сети (кроме самого себя – Zabbix server). Запустив скрипт, мы наблюдали процесс добавления в логах и на экране консоли.

В консольном выводе скрипта фиксировались сообщения о каждом добавлении. Например, для одного из устройств скрипт вывел: «Processing 172.16.224.79 -> group=Network Devices, templates=[‘10233’]». Такие сообщения позволяют администратору убедиться, что классификация произошла правильно.

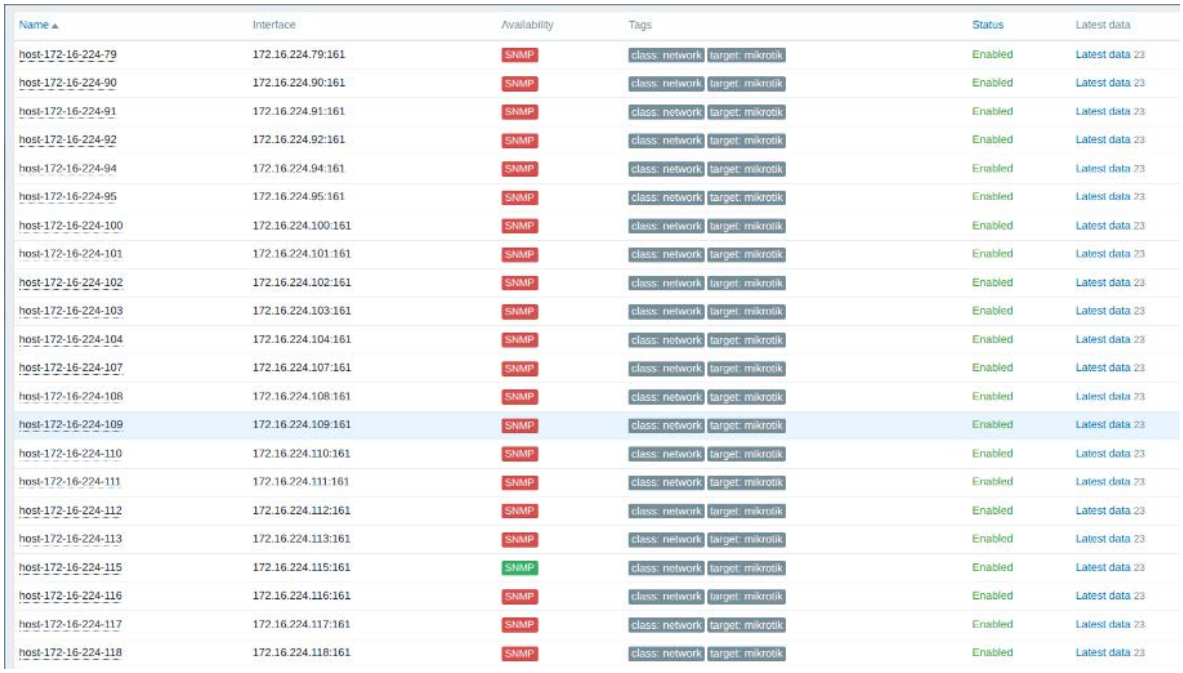
```
[+] Zabbix API URL: http://127.0.0.1:8080/api_jsonrpc.php
[+] Logging in to Zabbix...
[+] Logged in.
[+] Will scan: ['172.16.224.0/24']
[+] Scanning 172.16.224.0/24 ...
[+] Using interface: enx00e04c182c48


[i] Processing 172.16.224.79 / 18:fd:74:e0:67:cf -> group=Network Devices templates=['10233']
[+] Creating host host-172-16-224-79
[+] Creating SNMP host: host-172-16-224-79 (172.16.224.79) → group=Network Devices, templates=['10233']

[i] Processing 172.16.224.90 / 08:54:11:08:cf:a8 -> group=Network Devices templates=['10233']
[+] Creating host host-172-16-224-90
[+] Creating SNMP host: host-172-16-224-90 (172.16.224.90) → group=Network Devices, templates=['10233']

[i] Processing 172.16.224.91 / 08:54:11:31:cf:da -> group=Network Devices templates=['10233']
[+] Creating host host-172-16-224-91
[+] Creating SNMP host: host-172-16-224-91 (172.16.224.91) → group=Network Devices, templates=['10233']

[i] Processing 172.16.224.92 / bc:9b:5e:12:ea:1c -> group=Network Devices templates=['10233']
[+] Creating host host-172-16-224-92
[+] Creating SNMP host: host-172-16-224-92 (172.16.224.92) → group=Network Devices, templates=['10233']
```

Рисунок 2 – Фрагмент консольного вывода скрипта

На рисунке 2 представлен пример вывода скрипта в консоли. Скрипт последовательно обнаруживает новые узлы и сообщает о выполненных действиях: IP-адрес, MAC-адрес, а также назначенные группы и шаблон. Видно, что скрипт идентифицировал, например, узел с IP 172.16.224.79 как Network Device.

После выполнения скрипта в веб-интерфейсе Zabbix появились соответствующие записи. Результат в Zabbix: в разделе Configuration - Hosts отобразились новые хосты, каждому присвоены группы и шаблоны согласно настройке. Например, хост «172.16.224.79» имеет группы «Network» и «Mikrotik», и шаблон. Скриншот интерфейса Zabbix после добавления узлов (рис. 3) демонстрирует, что все хосты находятся в правильных группах, а их статус Enabled (включены) и данные опроса поступают. Ниже приведён этот скриншот.

В ходе эксперимента проверена работоспособность мониторинга на добавленных узлах: метрики CPU, памяти и ping начали поступать без дополнительных действий, что подтверждает успешное применение шаблонов. Также протестированы сценарии обновления: при повторном запуске скрипта спустя час добавленные ранее хосты не продублировались, так как скрипт распознал их как уже существующие в Zabbix по IP. Для отключения части узлов мы имитировали «падение» двух серверов – на дашборде Zabbix они корректно отметились как недоступные (триггеры Agent not available), что означает: добавленные хосты полноценно мониторятся системой.



Name	Interface	Availability	Tags	Status	Latest data
host-172-16-224-79	172.16.224.79:161	SNMP	class: network target: mikrotik	Enabled	Latest data 23
host-172-16-224-90	172.16.224.90:161	SNMP	class: network target: mikrotik	Enabled	Latest data 23
host-172-16-224-91	172.16.224.91:161	SNMP	class: network target: mikrotik	Enabled	Latest data 23
host-172-16-224-92	172.16.224.92:161	SNMP	class: network target: mikrotik	Enabled	Latest data 23
host-172-16-224-94	172.16.224.94:161	SNMP	class: network target: mikrotik	Enabled	Latest data 23
host-172-16-224-95	172.16.224.95:161	SNMP	class: network target: mikrotik	Enabled	Latest data 23
host-172-16-224-100	172.16.224.100:161	SNMP	class: network target: mikrotik	Enabled	Latest data 23
host-172-16-224-101	172.16.224.101:161	SNMP	class: network target: mikrotik	Enabled	Latest data 23
host-172-16-224-102	172.16.224.102:161	SNMP	class: network target: mikrotik	Enabled	Latest data 23
host-172-16-224-103	172.16.224.103:161	SNMP	class: network target: mikrotik	Enabled	Latest data 23
host-172-16-224-104	172.16.224.104:161	SNMP	class: network target: mikrotik	Enabled	Latest data 23
host-172-16-224-107	172.16.224.107:161	SNMP	class: network target: mikrotik	Enabled	Latest data 23
host-172-16-224-108	172.16.224.108:161	SNMP	class: network target: mikrotik	Enabled	Latest data 23
host-172-16-224-109	172.16.224.109:161	SNMP	class: network target: mikrotik	Enabled	Latest data 23
host-172-16-224-110	172.16.224.110:161	SNMP	class: network target: mikrotik	Enabled	Latest data 23
host-172-16-224-111	172.16.224.111:161	SNMP	class: network target: mikrotik	Enabled	Latest data 23
host-172-16-224-112	172.16.224.112:161	SNMP	class: network target: mikrotik	Enabled	Latest data 23
host-172-16-224-113	172.16.224.113:161	SNMP	class: network target: mikrotik	Enabled	Latest data 23
host-172-16-224-115	172.16.224.115:161	SNMP	class: network target: mikrotik	Enabled	Latest data 23
host-172-16-224-116	172.16.224.116:161	SNMP	class: network target: mikrotik	Enabled	Latest data 23
host-172-16-224-117	172.16.224.117:161	SNMP	class: network target: mikrotik	Enabled	Latest data 23
host-172-16-224-118	172.16.224.118:161	SNMP	class: network target: mikrotik	Enabled	Latest data 23

Рисунок 3 – Фрагмент интерфейса Zabbix после автоматического добавления хостов

Объём передаваемых данных. В логах Zabbix API зафиксировано 60 вызовов host.create в течение 2 минут. Это не создало заметной нагрузки: время выполнения каждого вызова 0,1 с, загрузка CPU на сервере возросла незначительно. Таким образом, можно предположить, что и при большем числе узлов, например, сотни скрипт справится, если вызывать API по очереди.

Вывод по эксперименту: Разработанный инструмент успешно автоматизировал ввод 60 устройств в мониторинг Zabbix, распределив их по категориям без ручного вмешательства. Все заявленные функции – определение типа, назначение групп и шаблонов, заполнение инвентаря – продемонстрированы на практике. Решение показало устойчивость: повторный запуск не дублирует хосты, а изменение состояний устройств отражается в Zabbix как обычно. Это подтверждает, что цель по автоматическому добавлению и сортировке хостов достигнута.

Заключение

В данной работе представлено решение для автоматического добавления хостов в систему мониторинга Zabbix и их интеллектуальной сортировки по группам на основе типа устройства. Проведен анализ стандартных подходов (автообнаружение и авторегистрация) и выявлено, что существуют сложности при их применении в крупномасштабных и разнородных сетях: высокая нагрузка при сканировании больших диапазонов, недостаток информации о добавляемых узлах и ограниченные возможности классификации.

Для устранения этих недостатков разработан внешний скрипт, который объединяет функциональность обнаружения и классификации. Скрипт использует протоколы ICMP и SNMP для выявления новых устройств, а также Zabbix API для их регистрации в системе с

автоматическим присвоением необходимых атрибутов (групп, шаблонов, инвентарных данных). Теоретический обзор архитектуры Zabbix и протоколов (Agent, SNMP, API) показал технологическую реализуемость такого подхода [2, 3].

Проведенное сравнение (таблица 1) демонстрирует, что предложенное решение обладает рядом преимуществ перед встроенными механизмами: оно более гибкое в настройке критериев, масштабируется под большие сети (за счёт разумного ограничения области сканирования) и обеспечивает богатые сведения о каждом добавленном хосте сразу при его появлении в мониторинге.

Результаты, отображенные в интерфейсе Zabbix, свидетельствуют о работоспособности решения. При этом вмешательство администратора свелось к минимуму – роль человека ограничилась наблюдением и настройкой базовых правил в скрипте.

Практическая ценность решения заключается в снижении трудозатрат на сопровождение мониторинга в динамично изменяющейся сети. При расширении инфраструктуры (добавлении новых устройств) система самостоятельно интегрирует их в мониторинг. Это уменьшает вероятность пропуска какого-либо узла, повышает актуальность базы мониторинга и скорость реакции на инциденты. Также исключается человеческий фактор ошибок при ручном вводе, например, опечатки в адресах, неверная группа и т.п.

Несмотря на все преимущества, предложенный подход требует поддержания актуальности правил в скрипте – при появлении нового типа устройств администратору нужно добавить соответствующее условие (например, если внедряется новое сетевое оборудование другого производителя, потребуется указать OID и шаблон для него). В перспективе этот процесс можно частично автоматизировать, используя базы данных MIB и шаблонов.

Представленные результаты позволяют заключить, что поставленная цель по созданию средства автоматического добавления и сортировки хостов в Zabbix достигнута. Разработанный скрипт можно внедрять в эксплуатацию для облегчения сопровождения мониторинга.

Литература

1. Gilis A. What is Zabbix? – TechTarget, 23.05.2024 [Электронный ресурс]. URL: https://www.techtarget.com/searchitoperations/definition/Zabbix (дата обращения: 15.09.2025).
2. Лифтинг Н., ван Бэкел Б. Zabbix 7: мониторинг ИТ-инфраструктуры. – СПб.: Питер, 2023. – 432 с.
3. Zabbix Team. Руководство пользователя Zabbix, версия 6.4. – URL: https://www.zabbix.com/documentation (дата обращения: 16.09.2025).
4. Javid H. SNMP: Управление и мониторинг сетей – Хабр. – 06.03.2025 [Электронный ресурс]. URL: https://habr.com/ru/articles/888692/ (дата обращения: 16.09.2025).
5. Прохоров А. Как мы делали мониторинг сети на 14 000 объектов – Хабр. 04.02.2019 [Электронный ресурс]. URL: https://habr.com/ru/companies/X5Tech/articles/438754/ (дата обращения: 20.09.2025).
6. Saiz M. Zabbix Auto-registration vs Auto-discovery – Clouding.io Knowledge Base, 23.04.2025 [Электронный ресурс]. URL: https://help.clouding.io/hc/en-us/articles/360010208340-Zabbix-Auto-registration-vs-Auto-discovery (дата обращения: 21.09.2025)
7. Bigelow S. Zabbix monitoring requires experience for best results – TechTarget, 09.07.2018 [Электронный ресурс]. URL: https://www.techtarget.com/searchitoperations/tip/Zabbix-monitoring-requires-experience-for-best-results (дата обращения: 25.09.2025)

8. Antonvn. Все, что вы хотели знать о MAC-адресе – Хабр, 13.01.2020 [Электронный ресурс]. URL: https://habr.com/ru/articles/483670/ (дата обращения: 25.09.2025)
9. What is ARP (Address Resolution Protocol)? – IONOS Digital Guide, 19.12.2017 [Электронный ресурс]. URL: https://www.ionos.com/digitalguide/server/know-how/what-is-arp-address-resolution-in-networks (дата обращения: 25.09.2025)
10. IEEE Registration Authority. Organizationally unique identifier (OUI) – официальный ресурс. – URL: https://standards.ieee.org/products-programs/regauth/oui/ (дата обращения: 07.10.2025).
11. Создание шаблона для Zabbix на примере DVR Trassir SDK – Хабр, 2018 [Электронный ресурс]. URL: https://habr.com/ru/articles/430534/ (дата обращения: 15.10.2025)
12. ТОП-6 фишек Zabbix: применение и настройка – Хабр, 27.02.2025 [Электронный ресурс]. URL: https://habr.com/ru/companies/banki/articles/886306/ (дата обращения: 23.10.2025).
13. Автоматическое переименование хостов в Zabbix – Хабр, 2010 [Электронный ресурс]. URL: https://habr.com/ru/articles/82465/ (дата обращения: 23.10.2025).
14. Корякина Ю. С., Айтбаев Ш. Т., Зимин И. В. Аналитический обзор методов реализации подсистем сбора, обработки, хранения и визуализации данных для SOC (Security Operations Center) // Проблемы автоматики и управления. — 2025. — № 2(53). — С. 29–40.