Fighting false positives and notification floods in Zabbix distributed environments

Nicola Mauri • Bitech SpA
A typical MSP scenario
Our scenario

Managed Services Provider

SERVER
ZABBIX

PROXY
ZABBIX

WAN

Customer 1

Customer 2

Customer 3
Zabbix in MSP scenarios

• Monitoring-as-a-Service (MaaS) model
• 1 Zabbix Server for several customers
• Active Proxies connecting through WAN
• Non-technical users receive alerts and need to take actions
Why fighting them?

• **False positives** undermine monitoring system reputation among your users.

• **Notification floods** make your monitoring system inefficient, often when you need it most.
How do you determine if a server host is alive?
Agent monitoring: the default way

![Diagram of Zabbix agent monitoring]

**Parent triggers**

**Template App Zabbix Agent**

- **Name**: Zabbix agent on \{HOST.NAME\} is unreachable for 5 minutes
- **Severity**: Not classified, Information, Warning, Average, High, Disaster
- **Expression**: \{host1:agent.ping.nodata(5m)\}=1
Agent monitoring: the default way

- Better than pinging: it catches a lot of problems with one single check.
- But it doesn't tell us which problem we have

<table>
<thead>
<tr>
<th>Condition</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host is down</td>
<td>Zabbix agent unreachable</td>
</tr>
<tr>
<td>Host is up, agent is down</td>
<td>Zabbix agent unreachable</td>
</tr>
<tr>
<td>Host is up, agent is unresponsive</td>
<td>Zabbix agent unreachable</td>
</tr>
</tbody>
</table>
Distributed environments

Expression: `{host1:agent.ping.nodata(5m)}=1`
Distributed environments

agent.ping.nodata()

also triggers upon non-agent problems:

• Zabbix Proxy is down
• Zabbix Server has connectivity issues
• Zabbix Server has performance issues
Solutions

• Triggers dependency?
• Implicit trigger dependency (ZBXNEXT-1891)
• Hosts dependency (Zabbix > 5.0)
• Event correlation?
Agent monitoring: a combined approach

Idea: using multiple checks to more precisely determine what's happening

- icmping
- net.tcp.service[tcp,,10050]
- agent.ping
Agent monitoring: a combined approach

Let’s put all these information together:

<table>
<thead>
<tr>
<th>ICMP</th>
<th>TCP 10050</th>
<th>agent.ping</th>
<th>Host status</th>
</tr>
</thead>
<tbody>
<tr>
<td>✗</td>
<td>✗</td>
<td></td>
<td>Host is unreachable</td>
</tr>
<tr>
<td>✔</td>
<td>✗</td>
<td></td>
<td>Agent service is down</td>
</tr>
<tr>
<td>✔</td>
<td>✔</td>
<td>nodata</td>
<td>System is overloaded or frozen</td>
</tr>
<tr>
<td>nodata</td>
<td>nodata</td>
<td>nodata</td>
<td>Unknown, probably not a host issue</td>
</tr>
<tr>
<td>✔</td>
<td>✔</td>
<td>✔</td>
<td>A great day for this agent</td>
</tr>
<tr>
<td>✗</td>
<td>✔</td>
<td>✔</td>
<td>Ok (a nasty network admin simply blocked our ICMPs)</td>
</tr>
</tbody>
</table>
Agent monitoring: a combined approach

Separate triggers allow sending notifications to the right people:

<table>
<thead>
<tr>
<th>Trigger Description</th>
<th>Responsible Teams</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host is unreachable</td>
<td>Networking Team</td>
</tr>
<tr>
<td>Agent service is down</td>
<td>Monitoring Team</td>
</tr>
<tr>
<td>System is overloaded or frozen</td>
<td>System Admins</td>
</tr>
<tr>
<td>Unknown, probably not an agent issue</td>
<td>none</td>
</tr>
<tr>
<td>A great day for this agent</td>
<td>–</td>
</tr>
<tr>
<td>Ok, a nasty network admin simply blocked our ICMPs</td>
<td>–</td>
</tr>
</tbody>
</table>
1. “Host is unreachable” trigger

<table>
<thead>
<tr>
<th>ICMP</th>
<th>TCP</th>
<th><code>agent.ping</code></th>
</tr>
</thead>
<tbody>
<tr>
<td>✘</td>
<td>✘</td>
<td>Host is unreachable</td>
</tr>
</tbody>
</table>

Trigger:

```
{host:icmpping.max(3m)}=0
and
{host:net.tcp.service[tcp,,10050].max(3m)}=0
```

On failures, `icmpping` and `net.tcp.service` return a '0' value, so you don’t need to use `nodata()`.
2. “Agent service is down” trigger

<table>
<thead>
<tr>
<th>ICMP</th>
<th>TCP</th>
<th>agent.ping</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔</td>
<td>✘</td>
<td></td>
</tr>
</tbody>
</table>

Agent service is down  
(host and its services may be up)

Trigger expression:

{host:net.tcp.service[tcp,,10050].max(30m)}=0

Depends on trigger:

"Host is unreacheable"
3. “System is overloaded or freezed”

<table>
<thead>
<tr>
<th>ICMP</th>
<th>TCP</th>
<th>agent.ping</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>✔</td>
<td>✔</td>
<td>nodata</td>
<td>System is overloaded or frozen</td>
</tr>
</tbody>
</table>

- We can connect to the agent TCP port
  → *host must be up, agent must be up.*

- We do have data for icmp item
  → *the proxy/server chain is working.*

- We have no data from agent
  → *the agent is unresponsive: the system must be blocked or overloaded.*
3. “System is overloaded or freezed”

<table>
<thead>
<tr>
<th>ICMP</th>
<th>TCP</th>
<th>agent.ping</th>
</tr>
</thead>
<tbody>
<tr>
<td>✔</td>
<td>✔</td>
<td>nodata</td>
</tr>
</tbody>
</table>

Problem expression:

\[
\{\text{host:net.tcp.service[tcp,,10050].min(1h)}\}=1 \\
\text{and}\ \{\text{host:icmpping.nodata(3h)}\}=0 \\
\text{and}\ \{\text{host:agent.ping.nodata(3h)}\}=1
\]

Recovery expression:

\[
\{\text{host:agent.ping.count(30m,1)}\}>10
\]

💡 Choose large time intervals to avoid flapping and some unobvious race conditions.
How do you detect Zabbix proxies failures?
Detecting active proxies failures

A remote **active** proxy often sits behind a firewall and doesn't accept direct connections from Zabbix Server.

If a proxy is unreachable, we simply observe a lack of data.
Detecting active proxies failures

Internal item: `zabbix[proxy,${PROXY_NAME},lastaccess]`
(not used in default templates)
“No connection from proxy” trigger

<table>
<thead>
<tr>
<th>lastaccess</th>
<th>Scenario</th>
</tr>
</thead>
<tbody>
<tr>
<td>&gt;1 min</td>
<td>No connection from proxy</td>
</tr>
</tbody>
</table>

Trigger expression:

```
{proxy1: zabbix[proxy, ${PROXY_NAME}, lastaccess].fuzzytime(1m)} = 0
```

It could mean:

- Proxy host is down
- Proxy service is stopped
- Proxy site is unreacheable (network or power outage)
Remote proxies: best practices

1. Keep the proxy host itself monitored, using agent.
2. `lastaccess` requires proxy name parameter. Use macros and ensure that item gets evaluated, i.e.:
   \[
   \{\text{myproxy1:zabbix[proxy,${PROXY\_NAME},lastaccess].nodata(24h)}\}=1
   \Rightarrow \text{“Cannot determine proxy last access time”}
   \]
3. Monitor proxy connectivity: ping its router from Zabbix Server, for rapid troubleshooting.
4. Use passive proxies, if security policies allow you.
How do you handle Zabbix Server connectivity issues?
Zabbix Server connectivity issues

No connection from proxy

No connection from proxy

No connection from proxy
Zabbix Server connectivity issues

Idea: auto-detecting connectivity problems and preventing alerts to be fired
Zabbix Server connectivity issues

Zabbix Server is network isolated!

\{zbxsrv:icmpping[\{\$WAN_GATEWAY\}].last()\}=0
and
\{zbxsrv:icmpping[\{\$WAN_HOST\}].last()\}=0
Zabbix Server connectivity issues

Make proxy trigger dependent on this trigger in order to avoid misleading notifications.

Template App Zabbix Proxy
No connection from proxy

Template App Zabbix Server
Zabbix Server is network isolated!

(In theory an isolated server should not be able to send notifications. However, mail notifications might be queued and sent later when connectivity is restored)
Conclusions

• Simple yet robust approach
• Tested on a wide range of deployments for 3+ years
• All logic stored into templates
  (no manual configuration or tuning needed)

Main benefits:

• More precise notifications
• Reduces false positives
• Prevents alert floods on distributed environments
Lessons learned in MSP environments
1. Do not overwhelm users

- PKI: Number of alerts /per user /per day
- Recommended (human-sustainable): \(< 20\)
2. Inform users about changes

- Users expect a **predictable and consistent behavior**
- Keep them informed about **changes** to monitoring service:
  - New hosts (auto-registration)
  - New discovered entities (LLD)
  - New triggers
  - New thresholds
3. Adopt a consistent severity classification

Applied thorough all your templates • Based on “objective” criteria • Associable to intervention priorities

For example:

<table>
<thead>
<tr>
<th>CRITERIA DEFINITION</th>
<th>SEVERITY</th>
<th>EXPECTED RESPONSE TIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data loss</td>
<td>Disaster</td>
<td>Immediately (too late?)</td>
</tr>
<tr>
<td>Service failure</td>
<td>High</td>
<td>Immediately</td>
</tr>
<tr>
<td>Imminent service failure</td>
<td>Average</td>
<td>As soon as possible</td>
</tr>
<tr>
<td>Unusual condition (no impact on services)</td>
<td>Warning</td>
<td>On daily/weekly basis</td>
</tr>
<tr>
<td>Non-problem event to keep track</td>
<td>Information</td>
<td>–</td>
</tr>
<tr>
<td>Internal monitoring event</td>
<td>Not classified</td>
<td>–</td>
</tr>
</tbody>
</table>
Thank you!

Nicola Mauri
Monitoring and Security Consultant

nm@nicolamauri.it
@nicolamauri

www.bitech.it