

# 17. 加密

## 概述

Zabbix支持使用传输层安全(TLS)协议v.1.2在Zabbix server、Zabbix proxy、Zabbix agent、zabbix\_sender和zabbix\_get程序之间的加密通信。从Zabbix 3.0开始支持加密，支持基于证书和共享密钥加密。

加密是可选的，可以针对各个组件进行配置（例如，一些proxies和agents可以配置为与服务器一起使用基于证书的加密，而其他可以使用共享密钥加密，剩下的可以像以前一样继续使用未加密的通信）。

服务器（代理）可以为不同的主机使用不同的加密配置。

Zabbix守护程序使用一个监听端口进行加密和未加密的传入连接。添加加密不需要在防火墙上打开新端口。

## 限制

- 私钥以明文形式存储在Zabbix组件启动期间可读的文件中。
- 共享密钥在Zabbix前端输入，并以纯文本形式存储在Zabbix数据库中。
- 内置加密不保护通讯：
  - 在运行Zabbix前端的Web服务器和用户Web浏览器之间
  - 在Zabbix前端和Zabbix服务器之间，
  - Zabbix服务器（代理）和Zabbix数据库之间。
- 目前，每个加密的连接都会打开一个完整的TLS握手，不会实现会话缓存和故障单。
- 根据网络延迟，添加加密会增加检查和操作的时间。  
例如，如果分组延迟为100ms则打开TCP连接并发送未加密的请求大约需要200ms，加密约1000毫秒添加建立TLS连接。  
可能需要增加超时，否则一些监控项和动作在agent运行远程脚本时，使用加密会失败，而不加密则成功。
- **网络发现**不支持加密。通过网络发现执行的Zabbix agent检查将是未加密的，如果Zabbix agent配置为拒绝未加密的连接，那么这种检查将不会成功。

## 用加密支持编译Zabbix

为了支持加密Zabbix必须编译并链接到三个加密库之一：

- *mbed TLS*（以前的*PolarSSL*）（版本1.3.9及更高版本1.3.x，*mbed TLS 2.x*当前不支持，它不是1.3分支的替代替代，Zabbix将不会使用*mbed TLS 2.x*进行编译。
- *GnuTLS*（3.1.18版）
- *OpenSSL*（1.0.1版）

通过指定“configure”脚本的选项来选择库：

- `--with-mbedtls[=DIR]`
- `--with-gnutls[=DIR]`
- `--with-openssl[=DIR]`

例如，要使用*OpenSSL*配置服务器和agent代理的源，可以使用以下内容：

```
./configure --enable-server --enable-agent --with-mysql --enable-ipv6 --with-net-snmp --with-libcurl --with-libxml2 --with-openssl
```

可以使用不同的加密库（例如具有*OpenSSL*的服务器，具有*GnuTLS*的agent代理）来编译不同的Zabbix组件。

如果您计划使用共享密钥[PSK]请考虑使用PSKs在Zabbix组件中使用*GnuTLS*或*mbed TLS*库。*GnuTLS*和*mbed TLS*库支持具有[Perfect Forward Secrecy](#)的PSK密码。*OpenSSL*库（版本1.0.1[1.0.2c]支持PSK[但可用的PSK密码套件不提供完美转发保密。

## 连接加密管理

Zabbix中的连接可以使用：

- 非加密（默认）
- [RSA certificate-based encryption](#)
- [基于RSA证书的加密](#)
- [PSK-based encryption](#)
- [基于PSK的加密](#)

有两个重要参数用于为Zabbix组件之间的连接指定加密：

- [TLSConnect](#)
- [TLSAccept](#)

[TLSConnect](#)指定要使用什么加密传出连接和可以采取3个中的一个[unencrypted][PSK][certificate][[TLSConnect](#)用于Zabbix proxy的配置文件（在主动模式下，仅指定与服务器的连接）和Zabbix agentd [用于主动检查）。在的zabbix前端的[TLSConnect](#)等效物是连接主机在字段配置→主机→<一些主机>→加密选项卡和连接代理字段中管理→代理→<一些代理>→加密选项卡。如果配置的连接加密类型失败，则不会尝试其他加密类型。

[TLSAccept](#)指定允许进入连接的连接类型。连接类型[unencrypted][PSK][certificate]可以指定一个或多个值。 [TLSAccept](#)用于Zabbix proxy的配置文件（在被动模式下，仅指定来自服务器的连接）和Zabbix agentd [用于被动检查）。在的zabbix前端的[TLSAccept](#)等效物是从主机连接在字段配置→主机→<一些主机>→加密选项卡和从连接代理在字段管理→代理→<一些代理>→加密选项卡。

通常，您仅为传入加密配置一种类型的加密。但您可能希望切换加密类型，例如从加密到基于证书的最小停机时间和回滚可能性。

要实现这一点，您可以[TLSAccept=unencrypted,cert](#)在agentd配置文件中设置并重新启动Zabbix agent [然后，您可以zabbix\_get使用证书测试与agent的连接。如果一切正常，你可以重新配置加密中的zabbix前端[agent配置→主机→<某些主机>→加密设置选项卡连接主机到“证书”。

当服务器配置缓存被更新（如果主机正在通过proxy进行监视时[proxy配置被更新），则与该agent的连接将被加密。

如果一切正常工作，您可以[TLSAccept=cert](#)在agent配置文件中设置并重新启动Zabbix agent [现在agent将只接受加密的基于证书的连接。未加密和基于PSK的连接将被拒绝

以类似的方式，它可以在服务器和proxy上运行。如果在Zabbix前端主机配置中连接设置为“证书”，则只能从agent [主动检查)和zabbix\_sender [trapper项目)接受基于证书的加密连接。

很可能您将配置传入和传出连接使用相同的加密类型或根本不加密。但从技术上讲，可以非对称地进行配置，例如基于传入和基于PSK的出口连接的基于证书的加密。

有关概述，每个主机的加密配置将显示在Zabbix前端配置→主机右侧的代理加密列中。配置显示示例：

例	连接到主机	允许从主机连接	拒绝从主机连接
<b>NONE</b>	未加密	未加密	加密证书和基于PSK的证书
<b>CERT NONE PSK CERT</b>	加密，基于证书	基于加密证书	未加密和基于PSK的
<b>PSK NONE PSK CERT</b>	加密，基于PSK的	加密PSK为主	未加密和基于证书
<b>PSK NONE PSK CERT</b>	加密，基于PSK的	未加密和基于PSK的加密	以证书为基础
<b>CERT NONE PSK CERT</b>	加密，基于证书	未加密□PSK或基于证书的加密	-

默认是未加密的连接。必须单独为每个主机和代理配置加密。

### zabbix\_get和zabbix\_sender加密

请参阅man-pages [zabbix\\_get](#)和[zabbix\\_sender](#)以使用加密。

### 密码套件

在Zabbix启动期间内部配置了密码套件，并且依赖于加密库，目前它们不是用户可配置的。

按照从高到低顺序的库类型配置密码：

密码库	证书密码	PSK密码
<i>mbedtls</i> TLS (PolarSSL) 1.3.9	TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256 TLS-ECDHE-RSA-WITH-AES-128-CBC-SHA256 TLS-ECDHE-RSA-WITH-AES-128-CBC-SHA TLS-RSA-WITH-AES-128-GCM-SHA256 TLS-RSA-WITH-AES-128-CBC-SHA256 TLS-RSA-WITH-AES-128-CBC-SHA	TLS-ECDHE-PSK-WITH-AES-128-CBC-SHA256 TLS-ECDHE-PSK-WITH-AES-128-CBC-SHA TLS-PSK-WITH-AES-128-GCM-SHA256 TLS-PSK-WITH-AES-128-CBC-SHA256 TLS-PSK-WITH-AES-128-CBC-SHA
<i>GnuTLS</i> 3.1.18	TLS_ECDHE_RSA_AES_128_GCM_SHA256 TLS_ECDHE_RSA_AES_128_CBC_SHA256 TLS_ECDHE_RSA_AES_128_CBC_SHA1 TLS_RSA_AES_128_GCM_SHA256 TLS_RSA_AES_128_CBC_SHA256 TLS_RSA_AES_128_CBC_SHA1	TLS_ECDHE_PSK_AES_128_CBC_SHA256 TLS_ECDHE_PSK_AES_128_CBC_SHA1 TLS_PSK_AES_128_GCM_SHA256 TLS_PSK_AES_128_CBC_SHA256 TLS_PSK_AES_128_CBC_SHA1
<i>OpenSSL</i> 1.0.2c	ECDHE-RSA-AES128-GCM-SHA256 ECDHE-RSA-AES128-SHA256 ECDHE-RSA-AES128-SHA AES128-GCM-SHA256 AES128-SHA256 AES128-SHA	PSK-AES128-CBC-SHA
<i>OpenSSL</i> 1.1.0	ECDHE-RSA-AES128-GCM-SHA256 ECDHE-RSA-AES128-SHA256 ECDHE-RSA-AES128-SHA AES128-GCM-SHA256 AES128-CCM8 AES128-CCM AES128-SHA256 AES128-SHA	ECDHE-PSK-AES128-CBC-SHA256 ECDHE-PSK-AES128-CBC-SHA PSK-AES128-GCM-SHA256 PSK-AES128-CCM8 PSK-AES128-CCM PSK-AES128-CBC-SHA256 PSK-AES128-CBC-SHA

密码套件使用证书:

TLS服务器			
TLS客户端	<i>mbed TLS (PolarSSL)</i>	<i>GnuTLS</i>	<i>OpenSSL 1.0.2</i>
<i>mbed TLS (PolarSSL)</i>	TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256	TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256	TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256
<i>GnuTLS</i>	TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256	TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256	TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256
<i>OpenSSL 1.0.2</i>	TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256	TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256	TLS-ECDHE-RSA-WITH-AES-128-GCM-SHA256

密码套件使用PSK

TLS服务器			
TLS客户端	<i>mbed TLS (PolarSSL)</i>	<i>GnuTLS</i>	<i>OpenSSL 1.0.2</i>
<i>mbed TLS (PolarSSL)</i>	TLS-ECDHE-PSK-WITH-AES-128-CBC-SHA256	TLS-ECDHE-PSK-WITH-AES-128-CBC-SHA256	TLS-PSK-WITH-AES-128-CBC-SHA
<i>GnuTLS</i>	TLS-ECDHE-PSK-WITH-AES-128-CBC-SHA256	TLS-ECDHE-PSK-WITH-AES-128-CBC-SHA256	TLS-PSK-WITH-AES-128-CBC-SHA
<i>OpenSSL 1.0.2</i>	TLS-PSK-WITH-AES-128-CBC-SHA	TLS-PSK-WITH-AES-128-CBC-SHA	TLS-PSK-WITH-AES-128-CBC-SHA

From: <https://www.zabbix.com/documentation/4.0/> - **Zabbix Documentation 4.0**

Permanent link: <https://www.zabbix.com/documentation/4.0/zh/manual/encryption>

Last update: **2018/10/24 09:12**

