

Mpp数据库安全性配置





安全性配置手册

这份手册描述了如何保护一个MPP数据库系统。这份手册假定读者有Linux/UNIX系统管理和数据库管理系统的知识。熟悉结构查询语言（SQL）也很有帮助。

因为MPP数据库是基于PostgreSQL 9.4的，这份手册假定读者对PostgreSQL比较熟悉。这份手册中通篇都为MPP数据库中类似的特性提供了对PostgreSQL文档的引用。 [PostgreSQL documentation](#)

这份手册为负责管理MPP数据库系统的系统管理员提供信息。

- **保护数据库**
介绍MPP数据库的安全性主题。
 - **MPP数据库端口和协议**
列出MPP集群内使用的网络端口和协议。
 - **配置客户端认证**
描述认证MPP数据库客户端的可用方法。
 - **配置数据库授权**
描述如何通过使用角色和权限在用户级别限制对数据库数据的授权访问。
 - **审计**
描述被记录并且应该被监控以检测安全性威胁的MPP数据库事件。
 - **加密数据和数据库连接**
描述如何在数据库中或者在网络上传输时加密数据以防止窃听攻击或者中间人攻击。
 - **安全性最佳实践**
这一章描述用户应该遵循的基本安全性最佳实践，它可以确保最高级别的系统安全性。
-

保护数据库

介绍MPP数据库的安全性主题。

安全性配置的目的是配置MPP数据库服务器以消除尽可能多的安全性缺陷。这份手册为最小安全性需求提供了底线，并且有额外的安全性文档对其进行补充。

根本的安全性需求有下面几类：

- [认证](#) 覆盖MPP数据库服务器支持并且用来建立客户端应用标识的机制。
- [授权](#) 与数据库用来授权客户端访问使用的特权和权限模型相关。
- [审计](#)，或者日志设置覆盖MPP数据库中跟踪成功或者失败的用户动作的可用日志选项。
- [数据加密](#) 阐述可用来保护静态数据以及传输中数据的加密功能。这包括与MPP数据库相关的安全性证书。

访问Kerberos化的Hadoop集群

用户可以使用MPP平台扩展框架（PXF）进行读写Hadoop文件系统的外部表。如果Hadoop集群受到Kerberos的保护（Kerberos化），MPP数据库必须被配置为允许外部表所有者用Kerberos认证。执行这种设置的步骤请见[为保护HDFS配置PXF](#)。

平台加固

平台加固涉及通过遵循最佳实践和实施联邦安全性标准来评估以及最小化系统弱点。加固产品基于US国防部的指导方针“安全性模板实现手册（STIG）”。加固工作移除不必要的包、禁用不需要的服务、设置严格的文件和目录权限、移除无主的文件和目录、为单用户模式执行认证并且为最终用户提供选项以配置服从最新STIG的包。

Parent topic: [安全性配置手册](#)

MPP数据库端口和协议

列出MPP集群内使用的网络端口和协议。

MPP数据库客户端用TCP在客户端连接端口（默认为5432）上连接到MPP的Master实例。这个监听端口可以在 `postgresql.conf` 配置文件中重新配置。客户端连接使用PostgreSQL的libpq API。psql命令行接口、一些MPP工具以及特定语言的编程API也将直接使用libpq库或者在内部实现libpq协议。

每一个Segment实例也有一个客户端连接端口，它被Master实例单独用来与Segment协调数据库操作。在MPP的Master上执行的 `gpstate -p` 命令，会列出MPP的Master以及主Segment和镜像Segment的端口分配。例如：

```
[gpadmin@mdw ~]$ gpstate -p
20190403:02:57:04:011030 gpstate:mdw:gpadmin-[INFO]:-Starting gpstate with args: -p
20190403:02:57:05:011030 gpstate:mdw:gpadmin-[INFO]:-local MPP Version: 'postgres (MPP
20190403:02:57:05:011030 gpstate:mdw:gpadmin-[INFO]:-master MPP Version: 'PostgreSQL &
20190403:02:57:05:011030 gpstate:mdw:gpadmin-[INFO]:-Obtaining Segment details from mas
20190403:02:57:05:011030 gpstate:mdw:gpadmin-[INFO]:--Master segment instance /data/ma
20190403:02:57:05:011030 gpstate:mdw:gpadmin-[INFO]:--Segment instance port assignments
20190403:02:57:05:011030 gpstate:mdw:gpadmin-[INFO]:-----
20190403:02:57:05:011030 gpstate:mdw:gpadmin-[INFO]:- Host      Datadir      Port
20190403:02:57:05:011030 gpstate:mdw:gpadmin-[INFO]:- sdw1    /data/primary/gpseg0  200
20190403:02:57:05:011030 gpstate:mdw:gpadmin-[INFO]:- sdw2    /data/mirror/gpseg0   2100
20190403:02:57:05:011030 gpstate:mdw:gpadmin-[INFO]:- sdw1    /data/primary/gpseg1  200
20190403:02:57:05:011030 gpstate:mdw:gpadmin-[INFO]:- sdw2    /data/mirror/gpseg1   2100
20190403:02:57:05:011030 gpstate:mdw:gpadmin-[INFO]:- sdw1    /data/primary/gpseg2  200
20190403:02:57:05:011030 gpstate:mdw:gpadmin-[INFO]:- sdw2    /data/mirror/gpseg2   2100
20190403:02:57:05:011030 gpstate:mdw:gpadmin-[INFO]:- sdw2    /data/primary/gpseg3  200
20190403:02:57:05:011030 gpstate:mdw:gpadmin-[INFO]:- sdw3    /data/mirror/gpseg3   2100
20190403:02:57:05:011030 gpstate:mdw:gpadmin-[INFO]:- sdw2    /data/primary/gpseg4  200
20190403:02:57:05:011030 gpstate:mdw:gpadmin-[INFO]:- sdw3    /data/mirror/gpseg4   2100
20190403:02:57:05:011030 gpstate:mdw:gpadmin-[INFO]:- sdw2    /data/primary/gpseg5  200
20190403:02:57:05:011030 gpstate:mdw:gpadmin-[INFO]:- sdw3    /data/mirror/gpseg5   2100
20190403:02:57:05:011030 gpstate:mdw:gpadmin-[INFO]:- sdw3    /data/primary/gpseg6  200
20190403:02:57:05:011030 gpstate:mdw:gpadmin-[INFO]:- sdw1    /data/mirror/gpseg6   2100
20190403:02:57:05:011030 gpstate:mdw:gpadmin-[INFO]:- sdw3    /data/primary/gpseg7  200
20190403:02:57:05:011030 gpstate:mdw:gpadmin-[INFO]:- sdw1    /data/mirror/gpseg7   2100
20190403:02:57:05:011030 gpstate:mdw:gpadmin-[INFO]:- sdw3    /data/primary/gpseg8  200
20190403:02:57:05:011030 gpstate:mdw:gpadmin-[INFO]:- sdw1    /data/mirror/gpseg8   2100
```

诸如后备复制、Segment镜像、统计信息收集以及Segment之间的数据交换等特性也会创建额外的MPP数据库网络连接。数据库启动时会创建一些持久连接而在查询执行之类的操作期间会创建临时连接。用于查询执行处理、数据移动以及统计信息收集的临时连接使用TCP和UDP协议在1025到65535范围内的可用端口。

Note: 为避免在初始化MPP数据库时MPP数据库与其他应用程序之间的端口冲突，请不要在操作系统参数net.ipv4.ip_local_port_range指定的范围内指定MPP数据库端口。例如，如果net.ipv4.ip_local_port_range = 10000 65535, 用户可以将MPP数据库基本端口号设置为该值以外的 范围：

```

PORT_BASE = 6000
MIRROR_PORT_BASE = 7000
REPLICATION_PORT_BASE = 8000
MIRROR_REPLICATION_PORT_BASE = 9000

```

用于MPP数据库的某些附加产品和服务有额外的网络需求。下面的表格列出了在MPP集群内使用的端口和协议，并且包括与MPP数据库集成的服务和应用。

Table 1. MPP数据库端口和协议

服务	协议/端口	描述
Master的SQL客户端连接	TCP 5432, libpq	MPP的Master主机上的SQL客户端连接端口。使用PostgreSQL的libpq API支持客户端。可配置。
Segment的SQL客户端连接	可变, libpq	Segment实例的SQL客户端连接端口。主机上的每一个主Segment和镜像Segment都必须有唯一的端口。端口在MPP系统初始化或者扩展时分配。gp_segment_configuration系统目录在port列中为每个Segment记录端口号。运行gpstate -p可以查看使用中的端口。
Segment镜像端口	可变, libpq	Segment从其主Segment接收镜像块的端口。该端口在镜像被设置时分配。端口号存储在gp_segment_configuration系统目录的mirror_port列中。
MPP数据库的Interconnect	UDP 1025-65535, 动态分配	在查询执行期间, Interconnect在MPP的Segment之间传输数据库元组。

服务	协议/端口	描述
后备Master的客户端监听器	TCP 5432, libpq	后备Master主机上的SQL客户端连接端口。通常和Master的客户端连接端口相同。可以用gpinitstandby工具的-P选项配置。
后备Master复制器	TCP 1025-65535, gpsyncmaster	Master主机上的gpsyncmaster进程会建立一个到第二Master主机的连接来把Master的日志复制到后备Master上。
MPP数据库文件装载和传输工具 : gpfdist、gpload、gptransfer	TCP 8080, HTTP TCP 9000, HTTPS	gpfdist文件服务工具能够在MPP主机或者外部主机上运行。在启动该服务器时用-p选项指定连接端口。 gpload和gptransfer工具会用一个配置文件中指定的端口或者端口范围运行一个或者更多gpfdist。
Gpperfmon代理	TCP 8888	执行在每一台MPP主机上的gpperfmon代理(gpmmmon和gpsmon)的连接端口。通过Master和Segment主机上postgresql.conf中的配置变量gpperfmon_port设置。
备份完成通知	TCP 25, TCP 587, SMTP	gpbackup备份工具可以选择在备份完成时向一个email地址列表发送邮件。SMTP服务必须在MPP的Master主机上被启用。

服务	协议/端口	描述
MPP数据库的安全shell (SSH) : gpssh、gpscp、gpssh-exkeys、gppkg、gpseg install	TCP 22, SSH	很多MPP的工具使用scp和ssh在主机之间传输文件并且管理集群中的MPP系统。
MPP平台扩展框架(PXF)	TCP 5888	PXF Java服务运行在每个MPP数据库segment主机上的5888端口。
Pgbouncer连接池	TCP, libpq	pgbouncer连接池运行在libpq客户端和MPP (或者PostgreSQL) 数据库之间。它可以运行在MPP的Master主机上, 但推荐将它运行在MPP集群之外的一台主机上。当它运行在一台单独的主机上时, pgbouncer可以当作MPP的Master主机的温备机制, 切换到MPP后备主机不要求重新配置客户端。在pgbouncer.ini配置文件中设置客户端连接端口和MPP的Master主机地址。

Parent topic: [安全性配置手册](#)

配置客户端认证

描述认证MPP数据库客户端的可用方法。

在MPP数据库系统第一次被初始化时, 系统含有一个预定义的超级用户角色。这个角色将具有与初始化该MPP数据库系统的操作系统用户相同的名称。这个

角色被称作gpadmin。默认情况下，系统被配置为只允许来自gpadmin角色的对数据库的本地连接。如果用户想要允许任

- [允许到MPP数据库的连接](#)
- [编辑pg_hba.conf文件](#)
- [认证方法](#)
- [限制并发连接](#)
- [加密客户端/服务器连接](#)

Parent topic: [安全性配置手册](#)

允许到MPP数据库的连接

客户端访问和认证受到配置文件pg_hba.conf（标准的PostgreSQL基于主机认证文件）的控制。有关该文件的详细信息请见PostgreSQL文档中的[pg_hba.conf文件](#)

在MPP数据库中，Master实例的pg_hba.conf文件控制着对MPP系统的客户端访问和认证。Segment也有自己的pg_hba.conf文件，但是它们已经被正确地配置为仅允许来自Master主机的客户端连接。Segment从不接受外部的客户端连接，因此没有必要修改Segment上的pg_hba.conf文件。

pg_hba.conf文件的一般格式是一组记录，每个记录一行。空行会被忽略，任何#号注释字符之后的文本也同样会被忽略。记录由若干个被空格或者制表符分隔的域构成。如果域值被加上引号，其中可以包含空格。记录不能跨行。每个远程客户端访问记录都是如下格式：

```
host database role address authentication-method
```

每个UNIX域套接字访问记录是如下格式：

```
local database role authentication-method
```

pg_hba.conf域的含义如下：

local

匹配使用UNIX域套接字的连接尝试。如果没有这类记录，就不允许UNIX域套接字连接。

host

匹配使用TCP/IP的连接尝试。除非服务器使用服务器配置参数listen_addresses的合适值启动，远程TCP/IP连接将不可能成功。

hostssl

匹配使用TCP/IP的连接尝试，但是只匹配使用SSL加密建立的连接。SSL必须在服务器启动时通过设置ssl配置参数来启用。对SSL认证的要求可以在postgresql.conf中配置。见[为SSL认证配置postgresql.conf](#)。

hostnossl

匹配使用TCP/IP但不用SSL的连接尝试。对SSL认证的要求可以在postgresql.conf中配置。见[为SSL认证配置postgresql.conf](#)。

database

指定这个记录匹配哪些数据库名。值all指定它匹配所有数据库。通过用逗号分隔数据库名，可以提供多个数据库名。可以指定一个包含数据库名的单独文件，方法是在这个域中给出加上前缀@的文件名。

role

指定这个记录匹配哪些数据库角色名。值all指定它匹配所有角色。如果指定的角色是一个组并且想包括其中的所有成员，可在角色名前面放一个+。通过用逗号分隔角色名，可以提供多个角色名。可以指定一个包含角色名的单独文件，方法是在这个域中给出加上前缀@的文件名。

address

指定这个记录匹配的客户端机器地址。该字段可以包含IP地址，IP地址范围或主机名。

IP地址范围是使用标准点分指定范围的起始地址，然后是斜杠 (/) 和CIDR掩码长度。掩码长度表明客户端IP地址必须匹配的高位二进制位数。给定IP地址中在这一长度右边的二进制位必须为零。在IP地址、/、CIDR掩码长度之间不能有任何空格。

IPv4地址范围的典型例子有 172.20.143.89/32 是单台主机, 或 172.20.143.0/24 是一个小型网络, 或 10.6.0.0/16 是一个大型网络. 对于单个主机, IPv6地址范围可能类似于::1/128 (在本例中为IPv6环回地址) 或fe80::7a31:c1ff:0000:0000/96用于小型网络。 0.0.0.0/0 代表所有IPv4地址, ::0/0 代表所有IPv6地址。要指定单台主机, 对于IPv4和IPv6分别使用CIDR掩码32和128.在网络地址中, 不要省略拖尾的零。

以IPv4格式给出的条目将仅匹配IPv4连接, 以IPv6格式给出的条目将仅匹配IPv6连接, 即使所表示的地址在IPv4-in-IPv6范围内。

Note: 如果主机系统C库不支持IPv6地址, 则IPv6格式的条目将被拒绝。如果指定了主机名 (将非IP地址或IP范围的地址视为主机名), 则将该名称与反向名称的结果进行比较客户端IP地址的分辨率 (例如, 如果使用DNS, 则为反向DNS查找)。主机名比较不区分大小写。如果匹配, 则转发名称对主机名执行解析 (例如, 正向DNS查找) 以检查其解析为的地址是否等于客户端IP地址。如果两个方向都匹配, 则认为该条目匹配。某些主机名数据库允许将IP地址与多个主机名相关联, 但是当要求系统解析IP时, 操作系统仅返回一个主机名地址。pg_hba.conf中使用的主机名必须是客户端IP地址的地址到名称解析返回的主机名, 否则行将不被视为匹配项。

在pg_hba.conf中指定了主机名时, 应确保名称解析相当快。设置本地名称解析缓存 (例如nscd) 可能是有利的。另外, 用户可以启用服务器配置参数log_hostname以查看客户端主机名而不是日志中的IP地址。

IP-地址

IP-掩码

这些域可以被用作CIDR-地址记法的替换选择。与指定掩码长度不同, 真实的掩码在一个单独的列中指定。例如, 255.0.0.0表示一个IPv4的CIDR掩码长度8, 而255.255.255.255表示CIDR掩码长度32。

authentication-method

指定连接时要使用的认证方法。详见 [认证方法](#)

CAUTION:

对于一个更安全的系统, 考虑从Master的pg_hba.conf中移除所有使用trust认证的连接。trust认证意味着不做任何认证就授予角色访问, 因此会绕过所有安全性。如果系统具有可用的ident服务, 可以把trust项替换为ident认证。

编辑pg_hba.conf文件

最初，pg_hba.conf文件被设置成gpadmin用户对数据库具有完全访问权限，而其他MPP Database角色则没有数据库访问权限。用户需要编辑pg_hba.conf文件，以使用户能够访问数据库并保证gpadmin用户的安全。需要考虑删除具有信任身份验证的条目，因为它们允许有权访问服务器的任何人以他们选择的任何角色进行连接。对于本地（UNIX套接字）连接，可使用ident身份验证，这要求操作系统用户匹配指定的角色。对于本地和远程TCP连接，身份验证要求客户端的主机运行ident服务。用户可以在主控主机上安装ident服务，然后对本地TCP连接使用ident身份验证，例如127.0.0.1/28。将身份验证用于远程TCP连接的安全性较差，因为它要求用户信任客户端主机上身份服务的完整性。

这个例子展示如何编辑Master的pg_hba.conf文件来允许对从所有角色访问所有数据库的远程客户端使用加密口令认证。

要编辑pg_hba.conf：

1. 在一个文本编辑器中打开文件\$MASTER_DATA_DIRECTORY/pg_hba.conf。
2. 为想要允许的每一类连接在文件中增加一行。记录会被顺序读取，因此记录的顺序是有意义的。通常，较早出现的记录将有比较严格的连接匹配参数以及较弱的认证方法，而较晚出现的记录将有较宽松的匹配参数和较强的认证方法。例如：

```
# 允许gpadmin用户使用ident认证本地访问所有数据库
local all gpadmin ident sameuser
host all gpadmin 127.0.0.1/32 ident
host all gpadmin ::1/128 ident
# 允许'dba'角色从任何具有IP地址192.168.x.x的主机访问任何数据库
# 并且是用md5加密的口令来认证用户
# 注意要使用SHA-256加密，用password替换下面行中的md5
host all dba 192.168.0.0/32 md5
```

认证方法

- [基本认证](#)
- [Kerberos认证](#)
- [LDAP认证](#)
- [SSL客户端认证](#)
- [基于PAM的认证](#)
- [Radius认证](#)

基本认证

支持下列基本认证方法：

Password 或者 MD5

要求客户端提供一个口令，可以为两者之一：

- Md5 – 口令作为MD5哈希传输。
- Password – 口令以明文传输。传输过程中应该总是使用SSL连接来防止口令嗅探。这是可配置的，更多信息请见*MPP数据库管理员手册*中的“加密口令”。

Reject

拒绝有匹配参数的连接。用户通常应该使用这种方法来限制来自特定主机或者不安全连接的访问。

Ident

基于客户端的操作系统用户名的认证。用户只对本地连接使用这种方法。对来自远程主机的TCP连接使用ident要求客户端的主机正在运行ident服务。身份认证方法仅应与受信任的封闭网络上的远程主机一起使用。

下面是pg_hba.conf基本认证项的一些例子： entries:

```
hostnossl all all 0.0.0.0 reject
hostssl all testuser 0.0.0.0/0 md5
local all gpuser ident
```

Kerberos认证

用户可以用一台Kerberos服务器（RFC 2743, 1964）认证。

pg_hba.conf 文件中Kerberos认证的格式是：

```
servicename/hostname@realm
```

可以把下列选项加到该项中：

Map

映射系统和数据库用户。

Include_realm

在ident映射文件中指定系统用户名中包括的realm名称的选项。

Krb_realm

为匹配的主体指定realm名称。

Krb_server_hostname

服务主体的主机名。

下面是一个pg_hba.conf的Kerberos项的例子：

```
host all all 0.0.0.0/0 krb5
      hostssl all all 0.0.0.0/0 krb5 map=krbmap
```

下列Kerberos服务器设置在postgresql.conf中被指定：

`krb_server_key file`

设置Kerberos服务器密钥文件的位置。

`krb_srvname string`

Kerberos服务名。

`krb_caseins_users boolean`

大小写敏感性。默认为off。

下面的客户端设置作为连接参数被指定：

`Krbsrvname`

要用于认证的Kerberos服务名。

LDAP认证

用户可以用一个LDAP目录认证。

- LDAPS和TLS上的LDAP选项可以加密到LDAP服务器的连接。
- 除非SSL被启用，从客户端到服务器的连接不会被加密。配置客户端连接以使用SSL加密来自该客户端的连接。
- 要配置或者自定义LDAP设置，将LDAPCONF 环境变量设置为到ldap.conf文件的路径并且将它加入到mpp_path.sh脚本。

下面是为用户系统配置LDAP认证的推荐步骤：

1. 用要通过LDAP认证的数据库用户/角色设置LDAP服务器。
 2. 在数据库上：
 - a. 验证要通过LDAP认证的数据库用户存在于数据库上。LDAP仅被用于验证用户名/口令对，因此角色应该存在于数据库中。
 - b. 更新\$MASTER_DATA_DIRECTORY中的pg_hba.conf文件为相应的用户使用LDAP作为认证方法。注意pg_hba.conf文件中第一个匹配用户/角色的项将被用作认证机制，因此项在该文件中的位置很重要。
 - c. 重载服务器让pg_hba.conf配置设置生效(gpstop -u)。
-

在auth-options中指定下面的参数。

ldapservers

要连接的LDAP服务器的名称或者IP地址。可以指定多个服务器，用空格分隔。

ldapprefix

在做简单绑定认证时，形成要绑定为的DN时自动加在用户名前面的字符串。

ldapsuffix

在做简单绑定认证时，形成要绑定为的DN时自动加在用户名后面的字符串。

ldapport

要连接的LDAP服务器上的端口号。如果没有指定端口，将使用LDAP库的默认端口设置。

ldaptls

设置为1让PostgreSQL和LDAP服务器之间的连接是用TLS加密。注意这只加密到LDAP服务器的流量 — 到客户端的连接仍将是未加密的（除非使用了SSL）。

ldapbasedn

在做搜索+绑定认证时，要开始在其中搜索用户的根DN。

ldapbinddn

在做搜索+绑定认证时，绑定到目录以执行搜索的用户的DN。

ldapbindpasswd

在做搜索+绑定认证时，绑定到目录以执行搜索的用户的口令。

ldapsearchattribute

在做搜索+绑定认证时，匹配正在搜索的用户名的属性。

Example:

```
ldapservers=ldap.mpp.com prefix="cn=" suffix=", dc=mpp, dc=com"
```

下面是LDAP认证的pg_hba.conf文件项的例子：

```
host all testuser 0.0.0.0/0 ldap ldap
ldapserver=ldapserver.mpp.com ldapport=389 ldapprefix="cn=" ldapsuffix=",ou=people,dc=mp
hostssl all ldaprole 0.0.0.0/0 ldap
ldapserver=ldapserver.mpp.com ldaptls=1 ldapprefix="cn=" ldapsuffix=",ou=people,dc=mpp,dc
```

SSL客户端认证

SSL认证将正在连接的客户端在SSL握手期间提供的SSL证书的公用名 (cn) 属性与被请求的数据库用户名进行比较。数据库用户应该在数据库中存在。映射文件可以被用来在系统用户名和数据库用户名之间进行映射。

SSL认证参数

认证方法：

- Cert

认证选项：

Hostssl

连接类型必须是hostssl。

map=*mapping*

映射。

这在pg_ident.conf文件中被指定，或者在ident_file服务器设置中指定的文件中指定。

下面是SSL客户端认证的pg_hba.conf项的例子：

```
Hostssl testdb certuser 192.168.0.0/16 cert
Hostssl testdb all 192.168.0.0/16 cert map=gpuser
```

OpenSSL配置

MPP数据库默认会读取\$GP_HOME/etc/openssl.cnf中指定的OpenSSL配置文件。用户可以通过修改或者更新这个文件并且重启服务器对OpenSSL的默认配置做更改。

创建一个自签名证书

自签名证书可以被用来测试，但是生产中应该使用由证书颁发机构（CA）（全球CA或者本地CA）签发的证书，这样客户端就能够验证服务器的身份。如果所有的客户端都是组织本地的，推荐使用本地CA。

要为服务器创建一个自签名证书：

1. 输入下列openssl命令： command:

```
openssl req -new -text -out server.req
```

2. 在提示中输入要求的信息。

确保为公共名输入本地主机名。挑战口令可以留空。

3. 这个程序生成受口令保护的密钥，它不接受短于四字符的口令。要移除口令（如果想让服务器自动启动还必须移除），运行下面的命令：

```
openssl rsa -in privkey.pem -out server.key rm privkey.pem
```

4. 输入旧口令解锁现有的密钥。然后输入下面的命令：

```
openssl req -x509 -in server.req -text -key server.key -out server.crt
```

这会把证书转变成一个自签名的证书并且把密钥和证书拷贝到服务器将查找它们的地方。

5. 最后，运行下列命令： `command:`

```
chmod og-rwx server.key
```

更多如何创建服务器私钥和证书的细节，请参考OpenSSL的文档。

为SSL认证配置postgresql.conf

下面的服务器设置需要在postgresql.conf配置文件中被指定：

- `ssl boolean`。 启用SSL连接。
- `ssl_renegotiation_limit integer`。 指定密钥重新协商之前的数据限制。
- `ssl_ciphers string`。 列出允许的SSL加密算法。

下列SSL服务器文件可以在Master的数据目录下找到：

- `server.crt`。 服务器证书。
- `server.key`。 服务器私钥。
- `root.crt`。 可信的证书机构。
- `root.crl`。 被证书机构撤销的证书。

配置SSL客户端连接

SSL选项：

`require`

仅使用SSL连接。如果存在一个根CA文件，以指定`verify-ca`时同样的方式验证证书。

`verify-ca`

仅使用SSL连接。验证服务器证书由一个可信的CA发出。

`verify-full`

仅使用SSL连接。验证服务器证书由一个可信的CA发出并且该服务器主机名匹配证书中的名称。

sslcert

客户端SSL证书的文件名。默认是~/.postgresql/postgresql.crt。

sslkey

用于客户端证书的密钥。默认是~/.postgresql/postgresql.key。

sslrootcert

包含SSL证书机构证书的文件名。默认是~/.postgresql/root.crt。

sslcr1

SSL证书撤销列表的名称。默认是~/.postgresql/root.crl。

客户端连接参数可以使用下列环境变量设置：

- sslmode – PGSSLMODE
- sslkey – PGSSLKEY
- sslrootcert – PGSSLROOTCERT
- sslcert – PGSSLCERT
- sslcr1 – PGSSLCRL

基于PAM的认证

“PAM”（可插拔认证模块）被用来验证用户名/口令对，类似于基本认证。PAM认证只在用户已经在数据库中存在时才有用。

参数

pamservice

默认PAM服务是postgresql。注意如果PAM被设置为读取/etc/shadow，认证将会失败因为PostgreSQL服务器由一个非根用户启动。

下面是PAM客户端认证的pg_hba.conf项的例子：

```
local all gpuser am pamservice=postgresql
```

Radius认证

RADIUS（远程认证拨号用户服务）认证通过向一台配置好的RADIUS服务器发送'Authenticate Only'类型的Access Request消息工作。它包括用于用户名、口令（加密）以及网络访问服务器（NAS）标识符的参数。这个请求使用由radiussecret选项中指定的共享秘密加密。RADIUS服务器会响应Access Accept或者Access Reject。

Note: 不支持RADIUS计费。

只有用户已经在数据库中存在时，RADIUS认证才有用。

为了强密码，RADIUS加密向量要求SSL被启用。

RADIUS认证选项

radiusserver

RADIUS服务器的名称。

radiussecret

RADIUS共享秘密。

radiusport

RADIUS服务器上要连接的端口。

radiusidentifier

RADIUS请求中的NAS标识符。

下面是RADIUS客户端认证的pg_hba.conf项的例子：

```
hostssl all all 0.0.0.0/0 radius radiusserver=servername radiussecret=sharedsecret
```

限制并发连接

要限制对MPP数据库系统的活动并发会话的数量，用户可以配置max_connections服务器配置参数。这是一个本地参数，意味着用户必须在Master、后备Master和每个Segment实例（主Segment和镜像Segment）的postgresql.conf文件中设置它。Segment上max_connections的值必须是Master上该值的5-10倍。

在用户设置max_connections时，还必须设置依赖参数max_prepared_transactions。在Master上，这个值必须被设置为至少和max_connections值一样大，而在Segment实例上应该设置为和Master上一样的值。

在\$MASTER_DATA_DIRECTORY/postgresql.conf（包括后备Master）中：

```
max_connections=100  
max_prepared_transactions=100
```

在所有Segment实例的SEGMENT_DATA_DIRECTORY/postgresql.conf中：

```
max_connections=500  
max_prepared_transactions=100
```

Note: 注意：调高这些参数的值可能会导致MPP数据库请求更多共享内存。为了缓和这种影响，考虑降低其他内存相关的参数，例如gp_cached_segworkers_threshold。

要更改允许的连接数：

1. 停止MPP数据库系统：

```
$ gpstop
```

2. 在Master主机上，编辑\$MASTER_DATA_DIRECTORY/postgresql.conf并且更改下面两个参数：
 - max_connections – 想要允许的活动用户会话数加上superuser_reserved_connections的数量
 - max_prepared_transactions – 必须大于等于 max_connections.
3. 在每个Segment实例上，编辑SEGMENT_DATA_DIRECTORY/postgresql.conf并且更改下面两个参数：
 - max_connections – 必须是Master上值的5-10倍。
 - max_prepared_transactions – 必须等于Master上的值。
4. 重启MPP数据库系统：

```
$ gpstart
```

加密客户端/服务器连接

MPP数据库对于客户端和Master服务器之间的SSL连接有原生支持。SSL连接可以防止第三方进行包嗅探，还能防止中间人攻击。只要客户端连接会通过不安全的链接，就应该使用SSL，并且只要使用客户端证书认证也应该使用SSL。

Note: 有关在gpfdist服务器和MPP数据库Segment主机之间加密数据的信息，请见[加密gpfdist连接](#).

启用SSL要求在客户端和Master服务器系统上都安装OpenSSL。可通过在Master的postgresql.conf中设置服务器配置参数ssl=on让MPP以启用SSL的方式启动。在以SSL模式启动时，服务器将在Master的数据目录中查找server.key（服务器私钥）文件和server.crt（服务器证书）文件。在启用SSL

的MPP系统能启动之前，这些文件必须被正确地设置好。

Important: 不要用口令保护私钥。服务器不会为私钥提示要求口令，而不提供口令数据库启动会失败并且报出错误。

自签名证书可以被用来测试，但是生产中应该使用由证书颁发机构（CA）（全球CA或者本地CA）签发的证书，这样客户端就能够验证服务器的身份。如果所有的客户端都是组织本地的，推荐使用本地CA。创建自签名证书的步骤请见 [创建一个自签名证书](#)

配置数据库授权

描述如何通过使用角色和权限在用户级别限制对数据库数据的授权访问。

Parent topic: [安全性配置手册](#)

访问权限和角色

MPP数据库使用角色管理数据库访问权限。角色的概念包括了用户和组的概念。一个角色可以是一个数据库用户、组或者两者皆有。角色可以拥有数据库对象（例如表）并且可以把那些对象上的特权指派给其他角色以控制对那些对象的访问。角色可以是其他角色的成员，因此成员角色可以继承其父角色的对象特权。

每一个MPP数据库系统都包含一组数据库角色（用户和组）。那些角色独立于服务器所运行的操作系统管理的用户和组。不过，为了方便用户可能想要维护操作系统用户名和MPP数据库角色名之间的关系，因为很多客户端应用使用当前

的操作系统用户名作为默认的数据库用户名。

在MPP数据库中，用户通过Master实例登入和连接，Master实例会验证它们的角色和访问特权。然后Master将在幕后用当前登入的角色向Segment实例发出命令。

角色被定义在系统层面上，因此它们对系统中的所有数据库都有效。

要让MPP数据库系统自举，刚刚初始化好的系统总是包含一个预定义的超级用户角色（也被称作系统该用户）。这个角色将与初始化MPP数据库系统的操作系统用户具有相同的名称。习惯上，这个角色被命名为gpadmin。要创建更多角色，用户首先必须作为这个初始角色连接。

管理对象特权

当一个对象（表、视图、序列、数据库、函数、语言、方案或表空间）被创建时，它会被指派一个拥有者。拥有者通常就是执行创建语句的角色。对于大部分种类的对象，初始状态只有拥有者（或者超级用户）可以对该对象做任何事情。要允许其他角色使用对象，必须授予特权。对每一类对象，MPP数据库支持下面的特权：

对象类型	特权
表、视图、序列	<ul style="list-style-type: none">• SELECT• INSERT• UPDATE• DELETE• RULE• ALL

对象类型	特权
外部表	<ul style="list-style-type: none"> • SELECT • RULE • ALL
数据库	<ul style="list-style-type: none"> • CONNECT • CREATE • TEMPORARY TEMP • ALL
函数	EXECUTE
Procedural Languages	USAGE
方案	<ul style="list-style-type: none"> • CREATE • USAGE • ALL

特权必须为每个对象单独授予。例如，在一个数据库上授予ALL并不会为该数据库中的对象授予完全的访问。它只授予所有的数据库级别特权（CONNECT, CREATE, TEMPORARY）给数据库本身。

使用SQL命令GRANT把对象上的特权给一个特定角色。例如：

```
=# GRANT INSERT ON mytable TO jsmith;
```

若要收回特权，可使用REVOKE命令。例如：

```
=# REVOKE ALL PRIVILEGES ON mytable FROM jsmith;
```

用户还可以使用DROP OWNED以及REASSIGN OWNED命令来管理弃用角色

所拥有的对象（注意：只有对象的拥有者或者超级用户可以删除一个对象或者重新指派拥有关系）。例如：

```
=# REASSIGN OWNED BY sally TO bob;  
      =# DROP OWNED BY visitor;
```

使用SHA-256加密

MPP数据库的访问控制大概能对应桔皮书的“C2”级安全性，而不是“B1”级。MPP数据库当前支持对象级别的访问特权。行级或者列级访问不被支持，标记安全性也不被支持。

行级和列级访问可以使用限制被选择的行列的视图来模拟。行级标签可以通过为表增加存储敏感度信息的额外列模拟，然后使用视图来控制基于该列的行级访问。然后可以为角色授予对视图的访问而不是对基表的访问。虽然这些变通方案不能提供和“B1”级安全性相同的安全性，但对于很多组织来说它们仍然是一种可行的替代方案。

要使用SHA-256加密，用户必须在系统或者会话级别上设置一个参数。这一节介绍如何使用一个服务器参数来实现SHA-256加密的口令存储。注意为了使用SHA-256加密进行存储，客户端认证方法必须被设置为password而不是默认的MD5（详见[配置SSL客户端连接](#)）。这意味着口令在网络上以明文方式传输，因此我们高度推荐用户设置SSL来加密客户端和服务器之间的通信信道。

用户可以在系统范围或者以会话为基础设置选择的加密方法。可用的加密方法是SHA-256以及MD5（为了向后兼容性）。

在系统范围设置加密方法

要在整个MPP系统（Master及其Segment）上设置password_hash_algorithm服

务器参数：

1. 作为超级用户登入MPP数据库实例。
2. 执行gpconfig把password_hash_algorithm设置为SHA-256：

```
$ gpconfig -c password_hash_algorithm -v 'SHA-256'
```

3. 验证设置：

```
$ gpconfig -s
```

将看到：

```
Master value: SHA-256  
Segment value: SHA-256
```

为单个会话设置加密方法

要为单个会话设置password_hash_algorithm服务器参数：

1. 作为超级用户登入MPP数据库实例。
2. 设置password_hash_algorithm为SHA-256：

```
# set password_hash_algorithm = 'SHA-256'
```

3. 验证设置：

```
# show password_hash_algorithm;
```

将会看到：

```
SHA-256
```

下面是展示新设置效果的例子：

1. 作为超级用户登入并且验证口令哈希算法设置：

```
# show password_hash_algorithm
password_hash_algorithm
-----
SHA-256
```

2. 创建一个带有口令且有登录特权的新角色。

```
create role testdb with password 'testdb12345#' LOGIN;
```

3. 更改客户端认证方法以允许SHA-256加密口令的存储：
在Master上打开pg_hba.conf文件并且加入下面的行：

```
host all testdb 0.0.0.0/0 password
```

4. 重启集群。
5. 作为刚创建好的用户testdb登入数据库。

```
psql -U testdb
```

6. 在提示下输入正确的口令。
 7. 验证口令被存储为SHA-256哈希。
口令的哈希被存储在pg_authid.rolpassword中。
-

8. 作为超级用户登入。

9. 执行下列查询：

```
# SELECT rolpassword FROM pg_authid WHERE rolname = 'testdb';  
Rolpassword  
-----  
sha256<64 hexadecimal characters>
```

用时间限制访问

MPP数据库让管理员能够限制角色在特定时间的访问。可使用CREATE ROLE或者ALTER ROLE命令指定基于时间的约束。

可以用日期或者日期和时间限制访问。无需删除和重建角色就可以移除这些约束。

基于时间的约束只适用于它们指派给的角色。如果角色是另一个有时间约束的角色的成员，时间约束不会被继承。

基于时间的约束仅在登录时被实施。SET ROLE以及SET SESSION AUTHORIZATION命令不受任何基于时间的约束的影响。

要为角色设置基于时间的约束，要求超级用户或者CREATEROLE特权。没有人可以为超级用户增加基于时间的约束。

有两种方法增加基于时间的约束。在CREATE ROLE或者ALTER ROLE命令中使用关键词DENY，后面接上下面的一种形式：

- 访问被限制的一个日子，以及可选的时间。例如，在周三不能访问。
 - 一个区间——也就是一个开始日期和结束日期以及可选的时间——在其间访问被限制。例如，从周三下午10点到周四上午8点期间不能访问。
-

用户可以指定多个限制，例如，周三的任何时间都不能访问并且在周五的下午3点到5点之间不能访问。

有两种方法指定一个日子。使用后面跟着带单引号的英语中平日术语的DAY或者0到6之间的一个数字，如下表所示。

英语术语	数字
DAY 'Sunday'	DAY 0
DAY 'Monday'	DAY 1
DAY 'Tuesday'	DAY 2
DAY 'Wednesday'	DAY 3
DAY 'Thursday'	DAY 4
DAY 'Friday'	DAY 5
DAY 'Saturday'	DAY 6

日子中的时间可以以12小时制或者24小时制指定。在词TIME后面接上带单引号的说明。只能指定小时和分钟并且用分号分隔 (:)。如果使用12小时制，在最后加上AM或者PM。下面的例子展示了多种时间说明。

```
TIME '14:00' # 24-hour time implied
TIME '02:00 PM' # 12-hour time specified by PM
TIME '02:00' # 24-hour time implied. This is equivalent to TIME '02:00 AM'.
```

Important: 基于时间的约束根据服务器时间实施。不考虑时区。

要指定一个时间区间，在其间访问被禁止，可以用词BETWEEN和AND指定两个日子/时间说明。如下所示。DAY总是需要的。

```
BETWEEN DAY 'Monday' AND DAY 'Tuesday'
```

```
BETWEEN DAY 'Monday' TIME '00:00' AND  
DAY 'Monday' TIME '01:00'
```

```
BETWEEN DAY 'Monday' TIME '12:00 AM' AND  
DAY 'Tuesday' TIME '02:00 AM'
```

```
BETWEEN DAY 'Monday' TIME '00:00' AND  
DAY 'Tuesday' TIME '02:00'  
DAY 2 TIME '02:00'
```

最后三个语句等效。

Note: 区间不能在周六之后回卷。

下面的语法不正确：

```
DENY BETWEEN DAY 'Saturday' AND DAY 'Sunday'
```

正确的说明是使用两个DENY子句，如下所示：

```
DENY DAY 'Saturday'  
DENY DAY 'Sunday'
```

下面的例子展示了创建带有基于时间约束的角色并且修改角色以增加基于时间的约束。只有基于时间约束所需的语句被显示。更多有关创建和修改角色的细节请见 *MPP数据库参考手册*中CREATE ROLE 和 ALTER ROLE的描述。

例 1 – 创建带有基于时间约束的新角色

在周末不允许访问。

```
CREATE ROLE generaluser  
DENY DAY 'Saturday'  
DENY DAY 'Sunday'  
...
```

例 2 – 修改角色以增加基于时间的约束

每晚的凌晨2点到4点之间不允许访问。

```
ALTER ROLE generaluser  
DENY BETWEEN DAY 'Monday' TIME '02:00' AND DAY 'Monday' TIME '04:00'  
DENY BETWEEN DAY 'Tuesday' TIME '02:00' AND DAY 'Tuesday' TIME '04:00'  
DENY BETWEEN DAY 'Wednesday' TIME '02:00' AND DAY 'Wednesday' TIME '04:00'  
DENY BETWEEN DAY 'Thursday' TIME '02:00' AND DAY 'Thursday' TIME '04:00'  
DENY BETWEEN DAY 'Friday' TIME '02:00' AND DAY 'Friday' TIME '04:00'  
DENY BETWEEN DAY 'Saturday' TIME '02:00' AND DAY 'Saturday' TIME '04:00'  
DENY BETWEEN DAY 'Sunday' TIME '02:00' AND DAY 'Sunday' TIME '04:00'  
...
```

例 3 – 修改角色以增加基于时间的约束

在周三或者周五下午的3点到5点间不允许访问。

```
ALTER ROLE generaluser  
DENY DAY 'Wednesday'  
DENY BETWEEN DAY 'Friday' TIME '15:00' AND DAY 'Friday' TIME '17:00'
```

删除基于时间的约束

要移除基于时间的约束，请使用ALTER ROLE命令。输入后面跟着要删除的日子/时间说明的关键词DROP DENY FOR。

```
DROP DENY FOR DAY 'Sunday'
```

任何含有DROP子句中全部或者部分条件的约束都会被移除。例如，如果一条现有的约束否定周一和周二的访问，并且DROP子句移除周一的约束，则这一条现有的约束会被完全删除。DROP子句会完全移除所有与DROP子句中约束交叠的约束。即便发生交叠的约束包含有比DROP子句更多的限制，它们也会被完全删除。

例 1 - 从一个角色移除一条基于时间的限制

```
ALTER ROLE generaluser  
DROP DENY FOR DAY 'Monday'  
...
```

这个语句将为例2中的generaluser角色移除所有与周一约束交叠的约束，即便其中有额外的约束。

审计

描述被记录并且应该被监控以检测安全性威胁的MPP数据库事件。

MPP数据库有能力审计很多事件，包括系统的启动和关闭、Segment数据库失效、导致错误的SQL语句以及所有的连接尝试和断开连接。MPP数据库还记录SQL语句和关于SQL语句的信息，并且可以以多种方式配置来记录不同细节的审

计信息。log_error_verbosity配置参数控制写入到服务器日志中的每一条消息的细节多少。类似地，log_min_error_statement参数允许管理员配置为SQL语句记录的细节层次，而log_statement参数确定要被审计的SQL语句的种类。当可审计事件由MPP数据库外部的主体发起时，MPP数据库会为所有可审计事件记录用户名。

MPP数据库通过只允许具有适当角色的管理员在日志文件上执行操作来防止对审计记录的未授权修改和删除。日志以一种使用逗号分隔值（CSV）的专有格式存储。每个Segment和Master都存储有自己的日志文件，不过这些都可以由管理员从远程访问。MPP数据库还通过log_truncate_on_rotation参数授权对旧日志文件的覆盖。这是一个本地参数并且必须在每个Segment和Master的配置文件中设置。

MPP提供了一个名为gp_toolkit的管理方案，用户可以用它来查询日志文件、系统目录和操作环境得到系统状态信息。包括用法在内的更多信息，请参考MPP数据库参考手册中的gp_toolkit管理方案附录。

查看数据库服务器的日志文件

MPP数据库中的每一个数据库实例（Master和Segment）都是一个运行着的PostgreSQL数据库服务器，它们都有自己的服务器日志文件。每天的日志文件被创建在Master和每个Segment的数据目录下的pg_log目录中。

服务器日志文件被写为逗号分隔值（CSV）格式。不是所有的日志项在所有的日志域中都有值。例如，只有与查询工作者进程相关的日志项才会有slice_id值。一个特定查询的相关日志项可以通过其会话标识符（gp_session_id）和命令标识符（gp_command_count）确定。

#	域名	数据类型	描述
1	event_time	timestamp with time zone	日志项被写到日志中的时间
2	user_name	varchar(100)	数据库用户名
3	database_name	varchar(100)	数据库名
4	process_id	varchar(10)	系统进程ID (带前缀"p")
5	thread_id	varchar(50)	线程计数 (带前缀"th")
6	remote_host	varchar(100)	在Master上, 是客户端机器的主机名/地址。在Segment上, 是Master的主机名/地址。
7	remote_port	varchar(10)	Segment或Master的端口号
8	session_start_time	timestamp with time zone	会话连接打开的时间

#	域名	数据类型	描述
9	transaction_id	int	Master上的顶层事务ID。这个ID是任何子事务的父级。
10	gp_session_id	text	会话标识符号（带前缀"con"）
11	gp_command_count	text	会话内部的命令编号（带前缀"cmd"）
12	gp_segment	text	Segment内容标识符（对主Segment带前缀"seg"，镜像Segment带前缀"mir"）。Master的内容id总是-1。
13	slice_id	text	切片id（查询计划被执行的部分）
14	distr_tranx_id	text	分布式事务ID
15	local_tranx_id	text	本地事务ID

#	域名	数据类型	描述
16	sub_tranx_id	text	子事务ID
17	event_severity	varchar(10)	值包括：LOG、ERROR、FATAL、PANIC、DEBUG1、DEBUG2
18	sql_state_code	varchar(10)	与日志消息相关的SQL状态代码
19	event_message	text	日志或者错误消息文本
20	event_detail	text	与错误或者警告消息相关的详细消息文本
21	event_hint	text	与错误或者警告消息相关的提示消息文本
22	internal_query	text	内部产生的查询文本
23	internal_query_pos	int	指向内部产生的查询文本中的光标

#	域名	数据类型	描述
24	event_context	text	产生消息的上下文
25	debug_query_string	text	带有完整细节的用户提供的查询字符串，用于调试。这个字符串可能会由于内部使用而修改。
26	error_cursor_pos	int	指向查询字符串中的光标
27	func_name	text	产生这个消息的函数
28	file_name	text	产生消息的内部代码文件
29	file_line	int	产生消息的内部代码文件的行号
30	stack_trace	text	与这个消息相关的栈跟踪文本

MPP提供一个名为gplogfilter的工具，它能被用来在一个MPP数据库日志文件中搜索匹配指定条件的项。这个工具默认会搜索位于默认日志位置的MPP的Master日志文件。例如，要显示Master日志文件的最后三行：

```
$ gplogfilter -n 3
```

用户还可以通过gpssh工具运行gplogfilter来立刻搜索所有的Segment日志文件。例如，要显示每个Segment日志文件的最后三行：

```
$ gpssh -f seg_host_file
=> source /usr/local/mpp-db/mpp_path.sh
=> gplogfilter -n 3 /gpdata/gp*/pg_log/gpdb*.csv
```

下面是MPP与安全相关的审计（或者日志）服务器配置参数，它们可以在postgresql.conf配置文件中设置：

域名	值范围	默认	描述
log_connections	Boolean	off	这会对服务器日志输出一行详细描述每个成功的连接。某些客户端程序（如psql）在决定是否要求口令时会尝试连接两次，因此重复的“connection received”消息并非总是表示问题。
log_disconnections	Boolean	off	在一个客户端会话终止时，这会在服务器日志中输出一行，其中会包括该会话的持续时间。

域名	值范围	默认	描述
log_statement	NONE DDL MOD ALL	ALL	控制那些SQL语句会被记录。DDL记录所有数据定义命令，如CREATE、ALTER和DROP命令。MOD记录所有DDL语句外加INSERT、UPDATE、DELETE、TRUNCATE以及COPY FROM。如果PREPARE和EXPLAIN ANALYZE语句中如果包含有适当类型的命令，它们也会被日志记录。
log_hostname	Boolean	off	连接日志消息默认只显示连接主机的IP地址。把这个选项打开会导致主机名也被记录。注意这依赖于用户的主机名解析设置，而且这有可能会带来不可忽视的性能损失。
log_duration	Boolean	off	致使每一个满足log_statement的完成语句的持续时间被记录。

域名	值范围	默认	描述
log_error_verbosity	TERSE DEFAULT VERBOSE	DEFAULT	为被记录的每条消息控制写入到服务器日志的细节多少。
log_min_duration_statement	number of milliseconds, 0, -1	-1	如果语句的持续时间大于等于指定的毫秒数，则在一个日志行中记录该语句和它的持续时间。将这个参数设置为0将打印出所有的语句及其持续时间。-1禁用这一特性。例如，如果用户将它设置为250，那么所有运行时间大于等于250ms的SQL语句将被记录。在跟踪应用中的未优化查询时，启用这一选项非常有用。

域名	值范围	默认	描述
log_min_messages	DEBUG5 DEBUG4 DEBUG3 DEBUG2 DEBUG1 INFO NOTICE WARNING ERROR LOG FATAL PANIC	NOTICE	控制哪些消息级别会被写入到服务器日志。每个级别包括其后的所有级别。级别越靠后，发送到日志的消息就越少。
log_rotation_size	0 - INT_MAX/1024 kilobytes	1048576	大于0时，将这个千字节数写入日志后，将创建一个新的日志文件。设置为零可禁用基于大小的新日志文件的创建。

域名	值范围	默认	描述
log_rotation_age	Any valid time expression (number and unit)	1d	决定个体日志文件的最大生存时间。在这个时间过去之后，一个新的日志文件将被创建。设置为零可禁用新日志文件基于时间创建。
log_statement_stats	Boolean	off	对每个查询，写入查询解析器、规划器和执行器的整体性能统计信息到服务器日志中。这是一种粗糙的画像手段。
log_truncate_on_rotation	Boolean	off	截断（重写）而不是追加到任何现存的同名日志文件。仅当一个新文件由于基于时间的轮转而被打开时，截断才会发生。例如，使用这个设置配合gpseg#-%H.log这样的log_filename会导致产生24个每小时的日志文件，然后循环地重写它们。关闭这一设置时，预先已经存在的文件在所有的情况下都会被追加内容。

Parent topic: [安全性配置手册](#)

加密数据和数据库连接

描述如何在数据库中或者在网络上传输时加密数据以防止窃听攻击或者中间人攻击。

- 客户端和Master数据库之间的连接可以用SSL加密。这可以用ssl服务器配置参数启用，该参数默认为off。将ssl参数设置为on允许客户端与Master的通信被加密。Master数据库必须为SSL设置好。更多有关用SSL加密客户端连接的信息请见[OpenSSL配置](#) OpenSSL配置。
- MPP数据库允许在MPP并行文件分发服务器、gpfdist以及Segment主机之间传输的数据的SSL加密。更多信息请见[加密gpfdist连接](#) 加密gpfdist连接。
- pgcrypto包中的加密/解密函数可以保护数据库中静止的数据。列级的加密可以保护敏感信息，例如社会保险号码或信用卡号。更多信息请见[用pgcrypto加密静止数据](#)用pgcrypto加密静止数据。

Parent topic: [安全性配置手册](#)

加密gpfdist连接

gpfdists协议是gpfdists协议的一个安全版本，它安全地标识文件服务器和MPP数据库并且加密它们之间的通信。使用gpfdists可以防止窃听攻击和中间人攻击。

gpfdists协议用下列显著的特性实现了客户端/服务器的SSL安全性：

- 要求客户端证书。
- 不支持多语言证书。
- 不支持证书撤销列表（CRL）。
- TLSv1协议被用于TLS_RSA_WITH_AES_128_CBC_SHA加密算法。这些SSL参数不能更改。
- 支持SSL重新协商。
- SSL的忽略主机失配参数被设置为假。
- gpfdist文件服务器（server.key）和MPP数据库（client.key）不支持含有口令的私钥。
- 发出适合于使用中的操作系统的证书是用户的责任。通常，支持将证书转换成必要的格式，例如使用<https://www.sslshopper.com/ssl-converter.html>上的SSL转换器。

用--ssl选项启动的gpfdist服务器只能用gpfdist协议通信。没有用--ssl选项启动的gpfdists服务器只能用gpfdists协议通信。更多gpfdists的细节请参考*MPP数据库管理员手册*。

有两种方法启用gpfdists协议：

- 用--ssl选项运行gpfdist然后在CREATE EXTERNAL TABLE语句的LOCATION子句中使用gpfdists协议。
- 使用一个SSL选项被设置为真的YAML控制文件并且运行gpload。运行gpload会用--ssl选项启动gpfdist服务器然后使用gpfdist协议。

在使用gpfdists时，下列客户端证书必须位于每一个Segment的\$PGDATA/gpfdists目录中：

- 客户端证书文件client.crt
- 客户端私钥文件client.key
- 可信证书机构root.crt

Important: 不要用口令保护私钥。服务器不会为私钥提示要求口令，因此数据装

载会在要求口令的情况下失败并且报出错误。

在使用带有SSL的gpload时，用户在YAML控制文件中指定服务器证书的位置。
在使用带有SSL的gpfdist时，用户用--ssl选项指定服务器证书的位置。

下面的例子展示如何安全地把数据装载到一个外部表中。这个例子从所有扩展名为txt的文件使用gpfdists协议创建了一个名为ext_expenses的可读外部表。这些文件被格式化为使用竖线 (|) 作为列定界符并且使用空格作为空值。

1. 在Segment主机上用--ssl选项运行gpfdist。

2. 登入数据库并执行下面的命令：

```
=# CREATE EXTERNAL TABLE ext_expenses  
  ( name text, date date, amount float4, category text, desc1 text )  
  LOCATION ('gpfdists://etlhost-1:8081/*.txt', 'gpfdists://etlhost-2:8082/*.txt')  
  FORMAT 'TEXT' ( DELIMITER '|' NULL ' ' );
```

用pgcrypto加密静止数据

MPP数据库的pgcrypto包提供加密静止在数据库中数据的函数。管理员可以加密有敏感信息的列已提供额外的保护层，例如社会保险号码或者信用卡号。存储为加密形式的数据库数据不能被不掌握加密密钥的用户读取，并且数据也无法直接从磁盘上读出。

当用户安装MPP数据库时，默认情况下会安装pgcrypto。用户必须在要使用该模块的每个数据库中显式启用pgcrypto。

pgcrypto允许使用对称和非对称加密的PGP加密。对称加密使用相同的密钥加密和解密数据，并且比非对称加密更快。在密钥交换不成为问题的环境中，对称加密是首选方法。对于非对称加密，公钥被用来加密数据而私钥被用来解密数据。这种方法比对称加密要慢一些并且要求更强的密钥。

使用pgcrypto总是会带来性能和可维护性方面的代价。有必要只对需要的数据使用加密。还有，记住无法通过索引来搜索加密数据。

在实现数据库内加密之前，考虑下列PGP限制。

- 不支持签名。这也意味着检查加密子密钥是否属于主密钥。
- 不支持加密密钥作为主密钥。通常不鼓励这种实践，因为这类限制不应该成为问题。
- 不支持多个子密钥。这可能看起来像是一个问题，因为这是常见的做法。在另一方面，用户不应将自己的常规GPG/PGP密钥用于pgcrypto，而是创建新的密钥，因为使用场景很不相同。

MPP数据库默认编译了zlib，这允许PGP加密函数在加密前压缩数据。在编译有OpenSSL时，将有更多算法可用。

因为pgcrypto函数在数据库服务器内部运行，数据和口令在pgcrypto和客户端应用之间以明文形式移动。为了最好的安全性，用户应该使用本地连接或者SSL连接并且用户应该信任系统管理员和数据库管理员。

pgcrypto会根据主PostgreSQL配置脚本的发现配置其自身。

当编译有zlib时，pgcrypto加密函数能够在加密前压缩数据。

pgcrypto有多种加密级别，范围从基本加密到高级内建函数。下面的表格显示所支持的加密算法。

Table 1. 表 1. pgcrypto支持的加密函数

值功能	内建	带有OpenSSL
MD5	yes	yes
SHA1	yes	yes
SHA224/256/384/512	yes	yes ¹ ...
其他摘要算法	no	yes ² ...

值功能	内建	带有 OpenSSL
Blowfish	yes	yes
AES	yes	yes ³
DES/3DES/CAST5	no	yes
Raw Encryption	yes	yes
PGP Symmetric-Key	yes	yes
PGP Public Key	yes	yes

创建PGP密钥

要在MPP数据库中使用PGP非对称加密，用户必须首先创建公私钥并且安装它们。

这一节假定用户正在一台Linux机器上用GNU隐私保护（gpg）命令行工具安装MPP数据库。使用最新版本的GPG来创建密钥。

从<https://www.gnupg.org/download/>为用户的操作系统下载和安装GNU隐私保护（GPG）。在GnuPG网站用户将找到流行的Linux发布的安装器以及Windows和Mac OS X安装器的链接。

1. 作为root，执行下面的命令并且从菜单中选择选项1：

```
# gpg --gen-key
gpg (GnuPG) 2.0.14; Copyright (C) 2009 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

gpg: directory '/root/.gnupg' created
gpg: new configuration file '/root/.gnupg/gpg.conf' created
gpg: WARNING: options in '/root/.gnupg/gpg.conf' are not yet active during this run
gpg: keyring '/root/.gnupg/secring.gpg' created
gpg: keyring '/root/.gnupg/pubring.gpg' created
Please select what kind of key you want:
(1) RSA and RSA (default)
(2) DSA and Elgamal
(3) DSA (sign only)
(4) RSA (sign only)
Your selection? 1
```

2. 如这个例子中所示，对提示做出响应并且遵照指示：

```
RSA keys may be between 1024 and 4096 bits long.
What keysize do you want? (2048) Press enter to accept default key size
Requested keysize is 2048 bits
Please specify how long the key should be valid.
0 = key does not expire
<n> = key expires in n days
<n>w = key expires in n weeks
<n>m = key expires in n months
<n>y = key expires in n years
Key is valid for? (0) 365
Key expires at Wed 13 Jan 2016 10:35:39 AM PST
Is this correct? (y/N) y
```

GnuPG needs to construct a user ID to identify your key.

```
Real name: John Doe
Email address: jdoe@email.com
Comment:
You selected this USER-ID:
"John Doe <jdoe@email.com>"
```

```
Change (N)ame, (C)omment, (E)mail or (O)kay/(Q)uit? Q
```

```
Change (r)name, (c)omment, (e)mail or (o)ray/(a)ut: ~
You need a Passphrase to protect your secret key.
(For this demo the passphrase is blank.)
can't connect to '/root/.gnupg/S.gpg-agent': No such file or directory
You don't want a passphrase - this is probably a *bad* idea!
I will do it anyway. You can change your passphrase at any time,
using this program with the option "--edit-key".
```

We need to generate a lot of random bytes. It is a good idea to perform some other action (type on the keyboard, move the mouse, utilize the disks) during the prime generation; this gives the random number generator a better chance to gain enough entropy.

We need to generate a lot of random bytes. It is a good idea to perform some other action (type on the keyboard, move the mouse, utilize the disks) during the prime generation; this gives the random number generator a better chance to gain enough entropy.

```
gpg: /root/.gnupg/trustdb.gpg: trustdb created
gpg: key 2027CC30 marked as ultimately trusted
public and secret key created and signed.
```

```
gpg: checking the trustdbgpg:
  3 marginal(s) needed, 1 complete(s) needed, PGP trust model
gpg: depth: 0 valid: 1 signed: 0 trust: 0-, 0q, 0n, 0m, 0f, 1u
gpg: next trustdb check due at 2016-01-13
pub  2048R/2027CC30 2015-01-13 [expires: 2016-01-13]
     Key fingerprint = 7EDA 6AD0 F5E0 400F 4D45  3259 077D 725E 2027 CC30
uid           John Doe <jdoe@email.com>
sub  2048R/4FD2EFBB 2015-01-13 [expires: 2016-01-13]
```

3. 通过输入下面的命令列出PGP密钥：

```
gpg --list-secret-keys
/root/.gnupg/secring.gpg
-----
sec  2048R/2027CC30 2015-01-13 [expires: 2016-01-13]
uid           John Doe <jdoe@email.com>
     ssb  2048R/4FD2EFBB 2015-01-13
```

2027CC30是公钥并且将被用于加密数据库中的数据。4FD2EFBB是私（秘密）钥并且将被用来解密数据。

4. 使用下面的命令导出密钥：

```
# gpg -a --export 4FD2EFBB > public.key  
# gpg -a --export-secret-keys 2027CC30 > secret.key
```

更多有关PGP加密函数的信息，请见[pgcrypto](#)的文档。

使用PGP加密表中的数据

这一节显示如何使用刚生成的PGP密钥加密插入到列中的数据。

1. 转储public.key文件的内容然后把它拷贝到剪切板：

```
# cat public.key
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v2.0.14 (GNU/Linux)

mQENBFS1Zf0BCADNw8Qvk1V1C36Kfcdw3Kpm/dijPfRyyEwB6PqKyA05jtWiXZTh
2His1ojSP6LI0cSkIqMU9LAINcecZhRIhBhuVgKIGSgd9texg2nnSL9Admqik/yX
R5syVKG+qcdWuvyZg9oOOmeyjhc3n+kbbRTEMuM3flbMs8shOwzMvstCUVmuHU/V
vG5rJAe8PuYDSJCJ74l6w7SOH3RiRlc7lFl6xYddV42l3ctd44bl8/i71hq2UyN2
/Hbsjii2ymg7ttw3jsWAX2gP9nssDgoy8QDy/o9nNqC8EGlig96ZFnFnE6Pwbhn+
ic8MD0IK5/GAIR6Hc0ZIHf8KEcavruQlikjnABEBAAG0HHRlc3Qga2V5IDx0ZXN0
a2V5QGgtYWIsLmNvbT6JAT4EEwECACGFAIS1Zf0CGwMFCQHhM4AGCwkIBwMCBhUI
AgkKCwQWAgMBAh4BAheAAoJEAd9cl4gJ8wwbfwH/3VyVsPkQl1owRjNxbvXGt1bY
7BfrvU52yk+PPZYoes9UpdL3CMRk8gAM9bx5Sk08q2UXSZLC6fFOpEW4uWgmGYf8
JRoc3ooezTkmcBW8l1bU0qGetzVxopdXLU PGCE7hVWQe9HcSntiTLxGov1mJAwO7
TAoccXLbyuZh9Rf5vLoQdKzcCyOHh5lqXaQOT100TeFeEpb9Tliwcntg3WCSU5P0
DGoUAOanjDZ3KE8Qp7V74fhG1EZVzHb8FajR62CXSHFKqpBgiNxnTOk45NbXADn4
eTUXPSnwPi46qoAp9UQogsfGyB1XDOTB2UOqhutAMECaM7VtpePv79i0Z/NfnBe5
AQ0EVLVI/QEIANabFdQ+8QMCAD0ipM1bF/JrQt3zUoc4BTqlCaxdyzAfz0tUSf/7
Zro2us99GIARqLWd8EqJcl/xmfcJiZyUam6ZAzzFXCgnH5Y1sdtMTJZdLp5WeOjw
gCWG/ZLu4wzxOFFzDkiPv9RDw6e5MNLtJrSp4hS5o2apKdbO4Ex83O4mJYnav/rE
iDDCWU4T0lhv3hSKCpke6LcwsX+7llozpaNmP0Ypwfi4hR3UUMP70+V1beFqW2J
bVLz3ILLouHRgpCzla+PzzbEKs16jq77vG9kqZTCIzXoWaLijuitRlfJkO3vQ9hO
v/8yAnkcAmowZrIBlyFg2KBzhunYmN2YvkUAEQEAAAYkBJQQYAQIADwUCVLVI/QIb
DAUJAeEzgAAKCRAHfXJelCfMMOHYCACFhInZA9uAM3TC44l+MrgMUJ3rW9izrO48
WrdTsxR8WkSNblxJoWnYxYuLyPb/shc9k65huw2SSDkj//0fRrl61FPHQNPSvz62
WH+N2lasoUaoJjb2kQGhLONFbJuevkyBylRz+hl/+8rJKcZOjQkmmK8Hkk8qb5x/
HMUc55H0g2qQAY0BpnJHgOOQ45Q6pk3G2/7Dbek5WJ6K1wUrFy51sNIGWE8pvgEx
/UUZB+dYqCwtvX0nnBu1KNCmk2AkEcFK3YoliCxonmdOxhFOv9AKjjojDyC65KJci
Pv2MikPS2fKOAgr1R3LpMa8zDEtI4w3vckPQNrQnNyuUtfj6ZoCxx
=XZ8J
-----END PGP PUBLIC KEY BLOCK-----
```

2. 创建一个名为userssn的表并且插入一些敏感数据，这个例子中是Bob和Alice的社会保险号码。在"dearmor("之后粘贴public.key的内容。

```
CREATE TABLE userssn( ssn_id SERIAL PRIMARY KEY,
  username varchar(100), ssn bytea);
```

```
INSERT INTO userssn(username, ssn)
```

```
SELECT robotccs.username, pgp_pub_encrypt(robotccs.ssn, keys.pubkey) AS ssn
FROM (
```

```
VALUES ('Alice', '123-45-6788'), ('Bob', '123-45-6799'))
```

```
AS robotccs(username, ssn)
```

```
CROSS JOIN (SELECT dearmor('-----BEGIN PGP PUBLIC KEY BLOCK-----
```

```
Version: GnuPG v2.0.14 (GNU/Linux)
```

```
mQENBFS1Zf0BCADNw8Qvk1V1C36Kfcwd3Kpm/dijPfRyyEwB6PqKyA05jtWiXZTh
2His1ojSP6LI0cSkIqMU9LAlncecZhRlhBhuVgKIGSgd9texg2nnSL9Admqik/yX
R5syVKG+qcdWuvyZg9oOOmeyjhc3n+kbbRTEMuM3flbMs8shOwzMvstCUVmuHU/V
vG5rJAe8PuYDSJCJ74l6w7SOH3RiRlc7lfl6xYddV42l3ctd44bl8/i71hq2UyN2
/Hbsjii2ymg7ttw3jsWAx2gP9nssDgoy8QDy/o9nNqC8EGlig96ZFnFnE6Pwbhn+
ic8MD0IK5/GAIR6Hc0ZIHf8KEcavruQlikjnABEBAAG0HHRlc3Qga2V5lDx0ZXN0
a2V5QGvtYWIsLmNvbT6JAT4EEwECACgFAIS1Zf0CGwMFCQHhM4AGCwkIBwMCBhUI
AgkKCwQWAgMBAh4BAheAAAoJEAd9cl4gJ8wwbfwH/3VyVsPkQl1owRjNxxvXGt1bY
7BfrvU52yk+PPZYoes9UpdL3CMRk8gAM9bx5Sk08q2UXSZLC6fFOpEW4uWgmGYf8
JRoc3ooezTkmcBW8l1bU0qGetzVxopdXLU PGCE7hVWQe9HcSntiTLxGov1mJAwO7
TAoccXLbyuZh9Rf5vLoQdKzcCyOHh5lqXaQOT100TeFeEpb9TIiwcntg3WCSU5P0
DGoUAOanjDZ3KE8Qp7V74fhG1EZVzHb8FajR62CXSHFKqpBgiNxnTOk45NbXADn4
eTUXPSnwPi46qoAp9UQogsfGyB1XDOTB2UOqhutAMECaM7VtpePv79i0Z/NfnBe5
AQ0EVLVI/QEIANabFdQ+8QMCAD0ipM1bF/JrQt3zUoc4BTqlCaxdyzAfz0tUSf/7
Zro2us99GIARqLWd8EqJcl/xmfcJiZyUam6ZAzzFXCgnH5Y1sdtMTJZdLp5WeOjw
gCWG/ZLu4wzxOFFzDkiPv9RDw6e5MNLtJrSp4hS5o2apKdbO4Ex83O4mJYnav/rE
iDDCWU4T0lhv3hSKCpke6LcwsX+7llozpaNmP0Ypwfi4hR3UUMP70+V1beFqW2J
bVLz3ILLouHRgpCzla+PzzbEKs16jq77vG9kqZTCIzXoWaLijuitRlfJkO3vQ9hO
v/8yAnkcAmowZrIBlyFg2KBzhunYmN2YvkUAEQEAAAYkBJQQYAQIADwUCVlVI/QIb
DAUJAeEzgAAKCRAhfXJelCfMMOHYCACFhInZA9uAM3TC44l+MrgMUJ3rW9izrO48
WrdTsxR8WkSNblxJoWnYxYuLyPb/shc9k65huw2SSDkj//0fRrl61FPHQNPSvz62
WH+N2lasoUaoJjb2kQGhLONfbJuevkyBylRz+hl/+8rJKcZOjQkmmK8Hkk8qb5x/
HMUc55H0g2qQAY0BpnJHgOOQ45Q6pk3G2/7Dbek5WJ6K1wUrFy51sNIGWE8pvgEx
/UUZB+dYqCwtvX0nnBu1KNCmk2AkEcFK3YoliCxomdOxhFOv9AKjjojDyC65KJci
Pv2MikPS2fKOA91R3LpMa8zDEtI4w3vckPQNrQnNyuUtfj6ZoCxx
=XZ8J
-----END PGP PUBLIC KEY BLOCK-----' AS pubkey) AS keys;
```

3. 验证ssn列被加密。


```
SELECT pgp_key_id(dearmor('-----BEGIN PGP PUBLIC KEY BLOCK-----  
Version: GnuPG v2.0.14 (GNU/Linux)
```

```
mQENBFS1Zf0BCADNw8Qvk1V1C36Kfcwd3Kpm/dijPfRyyEwB6PqKyA05jtWiXZTh  
2His1ojSP6LI0cSkIqMU9LAlncecZhRlhBhuVgKIGSgd9texg2nnSL9Admqik/yX  
R5syVKG+qcdWuvyZg9oOOmeyjhc3n+kbbRTEMuM3flbMs8shOwzMvstCUVmuHU/V  
vG5rJAe8PuYDSJCJ74l6w7SOH3RiRlc7lfl6xYddV42l3ctd44bl8/i71hq2UyN2  
/Hbsjii2ymg7ttw3jsWAx2gP9nssDgoy8QDy/o9nNqC8EGlig96ZFnFnE6Pwbhn+  
ic8MD0IK5/GAIR6Hc0ZIHf8KEcavruQlikjnABEBAAG0HHRlc3Qga2V5lDx0ZXN0  
a2V5QGvtYWIsLmNvbT6JAT4EEwECACgFAIS1Zf0CGwMFCQHhM4AGCwkIBwMCBhUI  
AgkKCwQWAgMBAh4BAheAAAOJEAAd9cl4gJ8wwbfwH/3VyVsPkQl1owRjNxxGt1bY  
7BfrvU52yk+PPZYoes9UpdL3CMRk8gAM9bx5Sk08q2UXSZLC6fFOpEW4uWgmGYf8  
JRoC3ooezTkmbCW8l1bU0qGetzVxopdXLUtPGCE7hVWQe9HcSntiTLxGov1mJAwO7  
TAoccXLbyuZh9Rf5vLoQdKzcCyOHh5lqXaQOT100TeFeEpb9Tliwcntg3WCSU5P0  
DGoUAOanjDZ3KE8Qp7V74fhG1EZVzHb8FajR62CXSHFKqpBgiNxnTOk45NbXADn4  
eTUXPSnwPi46qoAp9UQogsfGyB1XDOTB2UOqhutAMECaM7VtpePv79i0Z/NfnBe5  
AQ0EVLVI/QEIANabFdQ+8QMCADOipM1bF/JrQt3zUoc4BTqlCaxdyzAfz0tUSf/7  
Zro2us99GIARqLWd8EqJcl/xmfcJiZyUam6ZAzzFXCgnH5Y1sdtMTJZdLp5WeOjw  
gCWG/ZLu4wzxOFFzDkiPv9RDw6e5MNLtJrSp4hS5o2apKdbO4Ex83O4mJYnav/rE  
iDDCWU4T0lhv3hSKCpke6LcwsX+7liozp+aNmP0Ypwwfi4hR3UUMP70+V1beFqW2J  
bVLz3ILLouHRgpCzla+PzzbEKs16jq77vG9kqZTCIzXoWaLijuitRlfJkO3vQ9hO  
v/8yAnkcAmowZrIBlyFg2KBzhunYmN2YvkUAEQEAAAYkBJQQYAQIADwUCVLVI/QIb  
DAUJAEZgAAKCRAHfXJelCfMMOHYCACFhInZA9uAM3TC44l+MrgMUJ3rW9izrO48  
WrdTsxR8WkSNblxJoWnYxYuLyPb/shc9k65huw2SSDkj//0fRrl61FPHQNPsvz62  
WH+N2lasoUaoJjb2kQGhLONfbJuevkyBylRz+hl/+8rJKcZOjQkmmK8Hkk8qb5x/  
HMUc55H0g2qQAY0BpnJHgOOQ45Q6pk3G2/7Dbek5WJ6K1wUrFy51sNIGWE8pvgEx  
/UUZB+dYqCwtvX0nnBu1KNCmk2AkEcFK3YoliCxomdOxhFOv9AKjjojDyC65KJci  
Pv2MikPS2fKOAglR3LpMa8zDEtl4w3vckPQNrQNnYuUtfj6ZoCxxv  
=XZ8J  
-----END PGP PUBLIC KEY BLOCK-----');
```

```
pgp_key_id | 9D4D255F4FD2EFBB
```

这显示用于加密ssn列的PGP密钥ID是9D4D255F4FD2EFBB。每当新密钥被创建时推荐执行这一步，然后把ID存起来便于跟踪。

用户可以使用这个密钥来查看哪个密钥对被用来加密数据：

```
SELECT username, pgp_key_id(ssn) As key_used
FROM userssn;
username | Bob
key_used | 9D4D255F4FD2EFBB
-----+-----
username | Alice
key_used | 9D4D255F4FD2EFBB
```

Note: 不同的密钥可能有相同的ID。这很少见，但是是一种正常事件。客户端应用应该尝试用每一个来解密看看哪个适合 — 就像处理ANYKEY一样。请见pgcrypto文档中的[pgp_key_id\(\)](#)。

5. 使用私钥解密数据。

```
SELECT username, pgp_pub_decrypt(ssn, keys.privkey)
      AS decrypted_ssn FROM userssn
      CROSS JOIN
      (SELECT dearmor('-----BEGIN PGP PRIVATE KEY BLOCK-----
Version: GnuPG v2.0.14 (GNU/Linux)

IQOYBFS1Zf0BCADNw8Qvk1V1C36Kfcwd3Kpm/dijPfRyyEwB6PqKyA05jtWiXZTh
2His1ojSP6LI0cSkIqMU9LAIncecZhRIhBhuVgKIGSgd9texg2nnSL9Admqik/yX
R5syVKG+qcdWuvyZg9oOOmeyjhc3n+kkbRTEMuM3flbMs8shOwzMvstCUVmuHU/V
vG5rJAe8PuYDSJCJ74l6w7SOH3RiRlc7lFl6xYddV42l3ctd44bl8/i71hq2UyN2
/Hbsjii2ymg7ttw3jsWAx2gP9nssDgoy8QDy/o9nNqC8EGlig96ZFnFnE6Pwbhn+
ic8MD0IK5/GAIR6Hc0ZIHf8KEcavruQlikjnABEBAAEAB/wNfjvP1brRfjIm/j
XwUNm+sI4v2Ur7qZC94VTukPGf67lvqcYZJuqXxvZrZ8bl6mvI65xEUiZYy7BNA8
fe0PaM4Wy+Xr94Cz2bPbWgawnRNN3GAQy4rIBTrvqQWy+kmpbd87iTjwZidZNNmx
02iSzraq41Rt0Zx21Jh4rkpF67ftmzOH0vIrs0bWOvHUeMY7tCwmdPe9HbQeDIPr
n9CIIUqBn4/acTtCCIWAjREZn0zXAsNixtTIPC1V+9nO9YmecMkVwNfIPkIhymAM
OPFnuZ/Dz1rCRHjNHb5j6ZyUM5zDqUVnnezktxqrOENSxm0gfMGcpxHQogUMzb7c
6UyBBADSCXHPfo/VPVtMm5p1yGrNOR2jR2rUj9+poZzD2gjkt5G/xIKRlkB4uoQl
emu27wr9dVEX7ms0nvDq58iutbQ4d0JIDlchMeSRQZluErbIB75Vj3HtImblPjpn
4Jx6SWRXPUJPGXGI87u0UoBH0Lwij7M2PW7l1ao+MLEA9jAjQwQA+sr9BKPL4Ya2
r5nE72gsbCCLowkC0rdldf1RGtobwYDMpmYZhOaRKjkOTMG6rCXJxrf6LqiN8w/L
/gNziTmch35MCq/MZzA/bN4VMPyellwzxVZkJLsQ7yyqX/A7ac7B7DH0KfXciEXW
MSOAJhMmkIW1Q1RRNw3cnYi8w3q7X40EAL/w54FVvvPqp3+sCd86SAAapM4UO2R3
tIsuNVemMWdgNXwwK8AJsZ7VreVU5yZ4B8hvCuQj1C7geaN/LXhiT8foRsJC5o71
Bf+iHC/VNEv4k4uDb4lOgnHJYYyifB1wC+nn/EnXCZYQINMia1a4M6Vqc/RlIFTH4
nwkZt/89LsAiR/20HHRlc3Qga2V5IDx0ZXN0a2V5QGvtYWlsLmNvbT6JAT4EEwEC
```

```
ACgFAIS1Zf0CGwMFCQHhM4AGCwkIBwMCBhUIAgkKCwQWAgMBAh4BAheAAAoJEAd  
cl4gJ8wwbfwH/3VyVsPkQl1owRjNxxvXGt1bY7BfrvU52yk+PPZYoes9UpdL3CMRk  
8gAM9bx5Sk08q2UXSZLC6fFOpEW4uWgmGYf8JR0C3ooezTkmCBW8l1bU0qGetzVx  
opdXLuPGCE7hVWQe9HcSntiTLxGov1mJAwO7TAoccXLbyuZh9Rf5vLoQdKzcCyOH  
h5lqXaQOT100TeFeEpb9Tliwcntg3WCSU5P0DGoUAOanjDZ3KE8Qp7V74fhG1EZV  
zHb8FajR62CXSHFKqpBgiNxnTOk45NbXADn4eTUXPSnwPi46qoAp9UQogsfGyB1X  
DOTB2UOqhutAMECaM7VtpePv79i0Z/NfnBedA5gEVLVI/QEIANabFdQ+8QMCADOi  
pM1bF/JrQt3zUoc4BTqlCaxdyzAfz0tUSf/7Zro2us99GIARqLWd8EqJcl/xmfcJ  
iZyUam6ZAzzFXCgnH5Y1sdtMTJZdLp5WeOjwgCWG/ZLu4wzxOFFzDkiPv9RDw6e5  
MNLtJrSp4hS5o2apKdbO4Ex83O4mJYnav/rEiDDCWU4T0lhv3hSKCpke6LcwsX+7  
lioZp+aNmP0Ypwwfi4hR3UUMP70+V1beFqW2JbVLz3ILLouHRgpCzla+PzzbEKs16  
jq77vG9kqZTCIzXoWaLlJuitRlfJkO3vQ9hOv/8yAnkcAmowZrIBlyFg2KBzhunY  
mN2YvkUAEQEAAQAH/A7r4hDrnmzX3QU6FAzePIRB7niJtE2IEN8Auf05Q2PzKU/  
c1S72WjtqMAIAGYasDkOhfhcxanTneGuFVYggKT3eSDm1RFKpRjX22m0zKdwy67B  
Mu95V2OkIul6OCm8dO6+2fmkGxGqc4ZsKy+jQxtxK3HG9YxMC0dvA2v2C5N4TWi3  
Utc7zh//k6lbmaLd7F1d7DXt7Hn2Qsmo8l1rtgPE8grDToomTnRUodToyejEqKyI  
ORwsp8n8g2CSFaXSrEyU6HbFYXSxZealhQJGYLFOZdR0MzVtZQCn/7n+IHjupndC  
Nd2a8DVx3yQS3dAmvLzhFacZdjXi31wwj0moFOkEAOCz1E63SKNNksniQ11IRMJp  
gaov6Ux/zGLMstwTzNoul+Kr8/db0GISAy1Z3UoAB4tFQXEApOX9A4AJ2KqJqjOX  
cZVULenfDZaxrb9Lid7ZnTDXKVyGTWDF7ZHavHJ4981mCW17IU11zHBB9xMlx6p  
dhFvb0gdy0jSLaFMFr/JBAD0fz3RrhP7e6XII2zdBqGthjC5S/loKwwBgw6ri2yx  
LoxqBr2pl9PotJJ/JUMPhD/LxuTcOZtYjy8PKgm5jhnBDq3Ss0kNKAY1f5EkZG9a  
6l4iAX/NekqSyF+OgBfC9aCgS5RG8hYoOCbp8na5R3bgjuS8lzmVmm5OhZ4MDEwg  
nQP7BzmR0p5BahpZ8r3Ada7FcK+0ZLLRdLmOYF/yUrZ53SoYCZRzU/GmtQ7LkXBh  
Gjqied9Bs1MHdNUolq7GaexcjZmOWHEf6w9+9M4+vxtQq1nkIWqtaphewEmd5/nf  
EP3sIY0EAE3mmiLmHLqBju+UJKMNwFNeyMTqgcg50ISH8J9FRikBJQQYAQIADwUC  
VLVI/QIbDAUJAeEzgaAKCRAHfXJeICfMMOHYCACFhInZA9uAM3TC44I+MrgMUJ3r  
W9izrO48WrdTsxR8WkSNblxJoWnYxYuLyPb/shc9k65huw2SSDkj//0fRrl61FPH  
QNPSvz62WH+N2lasoUaoJjb2kQGhLonFbJuevkyByIRz+hl/+8rJKcZOjQkmmK8H  
kk8qb5x/HMUc55H0g2qQAY0BpnJHgOOQ45Q6pk3G2/7Dbek5WJ6K1wUrFy51sNIG  
WE8pvgEx/UUZB+dYqCwtvX0nnBu1KNCmk2AkEcFK3YoliCxomdOxhFOv9AKjjojd  
yC65KJciPv2MikPS2fKOAg1R3LpMa8zDEtI4w3vckPQNrQNnYuUtfj6ZoCxx  
=fa+6  
-----END PGP PRIVATE KEY BLOCK-----') AS privkey) AS keys;
```

```
username | decrypted_ssn
```

```
-----+-----
```

```
Alice | 123-45-6788
```

```
Bob | 123-45-6799
```

```
(2 rows)
```

如果用户创建了一个带有口令的密钥，用户可能不得不在这里输入它。不过对于这个例子的目的，口令为空。

密钥管理

根据用户是使用对称（单私钥）还是非对称（公私钥）加密算法，有必要安全地存放主密钥或私钥。存储加密密钥有很多种选项，例如放在文件系统中、密钥保险箱、加密USB、可信平台模块（TPM）或者硬件安全性模块（HSM）。

在规划密钥管理时，考虑下列问题：

- 密钥将被存储在哪里？
- 密钥应该何时过期？
- 如何保护密钥？
- 如何访问密钥？
- 密钥如何能被恢复和收回？

开放Web应用安全性项目（OWASP）为保护加密密钥提供了一份非常全面的手册。[guide to securing encryption keys](#)

¹ SHA2算法在版本0.9.8中被加入到OpenSSL，pgcrypto将使用内建代码。

² 任何OpenSSL支持的摘要算法都被自动的取用。但对加密不能这样，加密需要被明确地支持。

³ 从版本0.9.7开始OpenSSL中就包括了AES。对于较老的版本，pgcrypto将使用内建代码。

安全性最佳实践

这一章描述用户应该遵循的基本安全性最佳实践，它可以确保最高级别的系统安全性。

MPP数据库的默认安全性配置：

- 仅允许本地连接。
- 为超级用户（gpadmin）配置的基本认证。
- 超级用户被授权做任何事情。
- 只有数据库角色的口令被加密。

系统用户（gpadmin）

保护和限制对gpadmin系统用户的访问。

MPP要求一个UNIX用户ID来安装和初始化MPP数据库系统。这个系统用户在MPP文档中被称作gpadmin。gpadmin用户是MPP数据库中的默认数据库超级用户，也是MPP安装及其底层数据文件的文件系统所有者。默认的管理员账户是MPP数据库设计的根本。没有它系统无法运行，并且也没有办法限制gpadmin用户ID的访问。

这个gpadmin用户可以绕过MPP数据库的所有安全性特性。任何人通过这一用户ID登入到MPP主机，就可以读取、修改或者删除任何数据，包括系统目录数据和数据库访问权限。因此，非常有必要保护好gpadmin用户ID并且只允许必要的系统管理员可以接触到它。

只有在执行特定系统维护任务（例如升级或扩展）时，管理员才应该作为gpadmin登入到MPP。

数据库用户绝不应作为gpadmin登录，并且也绝不应以gpadmin运行ETL或者生产负载。

超级用户

被授予SUPERUSER属性的角色是超级用户。超级用户会绕过所有访问特权检查和资源队列。只有系统管理员应该被给予超级用户权利。

请见*MPP数据库管理员手册*中的“修改角色属性”。

登录用户

为每个登入系统的用户指派一个不同的角色并且设置LOGIN属性。

为了登录和审计的目的，每个被允许登入MPP数据库的用户应该被给予他们自己的数据库角色。对于应用或者Web服务，考虑为每一种应用或服务创建一个不同的角色。请见*MPP数据库管理员手册*中的“创建新角色（用户）”。

每一个登录角色应该被指派给一个单一的、非默认的资源队列。

组

使用组管理访问特权。

为每个对象/访问权限的逻辑分组创建一个组。

每个登录用户应该属于一个或者更多角色。使用GRANT语句给角色增加组访问。使用REVOKE语句从角色移除组访问。

不应该为组角色设置LOGIN属性。

见MPP数据库管理员手册中的“创建组（角色成员关系）”。

对象特权

只有拥有者和超级用户对新对象拥有完全的权限。权限必须被授予给其他角色（用户或组）以允许它们访问对象。每种类型的数据库对象都有不同的可以被授予的特权。使用GRANT语句给角色增加权限，使用REVOKE语句来移除权限。

用户可以使用REASSIGN OWNED BY语句更改对象的拥有者。例如，要准备删除一个角色，先要更改属于该角色的对象的拥有者。使用DROP OWNED BY删除角色拥有的对象，包括依赖对象。

方案可以被用于实施额外的对象权限检查层，但是方案权限不会覆盖包含在该方案中对象上设置的对象特权。

操作系统用户和文件系统

要防止网络入侵，系统管理员应该验证在组织内使用的口令足够强。下面的推荐可以强化口令：

- 最小口令强度推荐：至少9个字符。MD5口令应该为至少15个字符。
- 混合大小写字母。
- 混合字母和数字。
- 包括非字母数字字符。
- 选择一个用户可以记住的口令。

下面推荐了一些可以用来确定口令强度的口令破解软件。

- John The Ripper。一种快速灵活的口令破解程序。它允许使用多个单词列表
-

并且可以进行蛮力口令破解。该程序可以从<http://www.openwall.com/john/>得到。

- Crack。可能是最著名的口令破解软件，Crack也非常快，但是可能不如John The Ripper那么易用。该软件可以在<https://dropsafe.crypticide.com/alecm/software/crack/c50-faq.html>得到。

整个系统的安全性依赖于root口令的强度。该口令应该至少长达12个字符并且包括大写字母、小写字母、特殊字符和数字的组合。它不能基于任何词典中的词。

应该配置口令过期参数。

确保下面的行存在于文件/etc/libuser.conf的[import]小节中。

```
login_defs = /etc/login.defs
```

确保在[userdefaults]小节中没有以下列文本开头的行，因为这些词会覆盖来自/etc/login.defs的设置：

- LU_SHADOWMAX
- LU_SHADOWMIN
- LU_SHADOWWARNING

确保下面的命令不会产生输出。通过这一命令列出的任何账号都应该被锁定。

```
grep "^+:" /etc/passwd /etc/shadow /etc/group
```

注意：我们强烈推荐客户在初始设置后更改他们的口令。

```
cd /etc
chown root:root passwd shadow group gshadow
chmod 644 passwd group
chmod 400 shadow gshadow
```

找出所有全域可写的文件以及没有设置其粘滞位的文件。

```
find / -xdev -type d \( -perm -0002 -a ! -perm -1000 \) -print
```

为前一个命令结果中的所有目录设置其粘滞位（# `chmod +t {dir}`）。

找出所有全域可写的文件并且修正每一个被列出的文件。

```
find / -xdev -type f -perm -0002 -print
```

为前述命令给出的所有文件设置正确的权限（# `chmod o-w {file}`）。

找出所有不属于有效用户或组的文件，然后为它们分配一个拥有者或者移除文件。

```
find / -xdev \( -nouser -o -nogroup \) -print
```

找出所有全域可写的目录并且确认它们属于root或者某个系统账户（假定只有系统账户的用户ID低于500）。如果该命令生成输出，验证其分配是否正确或者将它们重新分配给root。

```
find / -xdev -type d -perm -0002 -uid +500 -print
```

口令质量、口令过期策略、口令重用、口令重试尝试以及更多认证设置可以通过可插拔认证模块（PAM）框架配置。PAM在目录/etc/pam.d中查找应用相关的配置信息。运行`authconfig`或者`system-config-authentication`将重写PAM配置文件，这会毁掉任何手工更改并且将它们替换为系统默认配置。

默认的PAM模块`pam_cracklib`提供了口令的强度检查。要配置`pam_cracklib`以要求至少一个大写字符、小写字符、数字和特殊字符（U.S.国防部指导方针推荐），可编辑文件/etc/pam.d/system-auth并且在对应于口令前置条件`pam_cracklib.so try_first_pass`的行中包括下列参数。

```
retry=3:  
dcredit=-1. Require at least one digit  
ucredit=-1. Require at least one upper case character  
ocredit=-1. Require at least one special character  
lcredit=-1. Require at least one lower case character  
minlen=14. Require a minimum password length of 14.
```

例如：

```
password required pam_cracklib.so try_first_pass retry=3\minlen=14 dcredit=-1 ucredit=-1 ocre
```

可以设置这些参数来反映用户的安全性策略需求。注意口令限制不适用于root口令。

PAM模块pam_tally2提供了在指定失次数的败登录尝试之后锁住用户账户的功能。要实施口令封锁，可编辑文件/etc/pam.d/system-auth来包括下列行：

- 第一个认证行应该包括：

```
auth required pam_tally2.so deny=5 onerr=fail unlock_time=900
```

- 第一个账户行应该包括：

```
account required pam_tally2.so
```

这里，deny参数被设置以限制重试次数为5并且unlock_time已经被设置为900秒来保持账户在被解锁前锁定900秒。请配置这些参数以反映用户的安全性策略需求。被锁定的账户可以用pam_tally2工具手工解锁：

```
/sbin/pam_tally2 --user {username} -reset
```

用户可以使用PAM限制重用最近用过的口令。可以设置pam_unix模块的remember选项来记住最近的口令并且阻止重用它们。要做到这一点，可

在/etc/pam.d/system-auth中编辑适当的行来包括remember选项。

例如：

```
password sufficient pam_unix.so [ ... existing_options ...]  
remember=5
```

用户可以设置要记住的历史口令的数量以正确地反映其安全性策略需求。

```
cd /etc  
chown root:root passwd shadow group gshadow  
chmod 644 passwd group  
chmod 400 shadow gshadow
```

Parent topic: [安全性配置手册](#)