Redis安全注意事项

本文简要介绍 Redis 安全相关的话题, 包括: Redis访问控制机制、Redis源码安全性, 外部输入可能触发的恶意攻击, 以及其他相关问题。

如果要反馈安全问题, 请到 GitHub 上提出 issue, 当然, 如果需要私密通信, 请使用文末提供的 GPG key。

Redis 总体安全模型

Redis 设想的运行环境, 是与受信客户端在内网中进行通讯. 也就是说, Redis 实例不应该直接暴露到公网上, 也不应该让不受信的客户机直连到 Redis 的 TCP端口/或UNIX socket。

比如, 可以将 Redis 作为web系统的 database, cache, 或者 messaging system。 用户只能与web进行交互, 由WEB应用来进行查询或执行其他操作。

在这种情况下, web应用作为桥头堡, 连接 Redis, 避免Redis与不受信任的客户端(如浏览器)进行直接交互。

这只是一中特定场景, 但总体说来, 不受信任的客户端与 Redis 之间, 必须有一层 ACL(访问控制层)实现, 用于鉴权和校验用户输入, 并决定是否对 Redis 实例执行操作。

总的来说, Redis 并没有为安全问题做过多设计, 最主要的原因是为了保证高性能, 以及使用简便。

网络安全问题

除了受信网络的客户端, 其他客户端发起的网络请求需要被拦截, 所以运行 Redis 服务的系统, 应该只允许使用 Redis 的那些应用程序直连。

如果使用的是 Linux虚拟机(Linode, EC2, 等等), 因为这些机器可能直接暴露在公网上, 所以需要防火墙来保护 Redis 端口, 阻止外部访问。而本地的客户端则通过回环地址(loopback, 127...*)来访问Redis。

当然, 也可以将 Redis 端口直接绑定到本机的某块网卡/IP上, 在 **redis.conf** 配置文件中增加如下配置即可:

```
bind 127.0.0.1
```

假如不对外部访问做隔离, 可能会带来严重的安全隐患. 例如, 攻击者只要执行一个 **FLUSHALL** 命令, 就能让Redis的所有数据Over。

保护模式(Protected mode)

杯具的是, 很多 Redis 实例都没有拒绝外部网络访问。很多具有公网IP的实例就这样暴露了. 基于这种情况, 从 Redis 3.2.0 版本开始, 如果使用默认配置(绑定到所有网卡), 并且没有密码验证保护, 则会进入一种特殊的模式, 称为**保护模式(Protected mode)**. 在保护模式下, Redis 只允许本地回环地址访问。 其他地址的客户端在连接时, 会收到错误信息, 其中描述了具体原因以及如何配置。

我们希望保护模式能有效降低 Redis 实例的安全问题, 当然, 系统管理员也可以禁用保护模式, 或者手动绑定到所有IP。

身份验证功能

虽然 Redis 没有实现访问控制, 但也提供了一个小小的 身份验证层(authorization layer), 通过 **redis.conf** 文件来开启。

如果启用了 身份验证, Redis 会拒绝所有未经身份验证的客户端请求。客户端可以通过 **AUTH** 命令+密码 的方式执行身份验证。

因为密码以明文的方式设置在 redis.conf 文件中。所以密码应该足够长, 以防止暴力破解, 原因如下:

- Redis 的查询效率非常高。客户端每秒可以执行很多次密码验证。
- 密码存储在 **redis.conf** 中, 所以管理员不用记忆密码, 需要的时候直接拷贝即可, 因此设置很长的密码并没有什么影响。

身份验证层的目的, 是提供可选的一个冗余层. 如果防火墙失效, 或者未能有效保护 Redis, 只要外部人员不知道密码, 依然不能访问 Redis 实例。

和其他 Redis 命令一样, `AUTH` 命令也是不进行加密传输的, 所以不能防止网络窃听, 假若网络情况特别不安全的话。

数据加密支持

因为 Redis 不支持加密. 想要在互联网/或不可信网络上实现加密传输, 就需要额外的保护层, 例如SSL代理。我们推荐使用 spiped。

禁用特定命令

可以将某些 Redis 命令禁用, 或者重命名, 这样一般的客户端就不能执行某些危险的命令了。

例如, 虚拟服务提供商, 可能会同时给客户提供 Redis 实例管理服务. 这时候, 就不允许客户自己调用 **CONFIG** 命令来修改实例配置, 但服务提供商自己应该能够创建和销毁这些实例。

这种情况下, 可以通过重命名, 或者在命令表中隐藏这些命令。该特性可以通过 redis.conf 配置文件指定. 如:

```
rename-command CONFIG b840fc02d524045429941cc15f59e41cb7be6c52
```

此处将 **CONFIG** 命令重命名为另一个非常复杂的名字. 当然, 也可以将命令重命名为空串 `""`, 以禁用某些命令:

```
rename-command CONFIG ""
```

精心构造的外部输入攻击

即使黑客没有密码, 也可能攻击到 Redis. 例如, 黑客有可能利用 Redis内部算法和数据结构的漏洞, 最坏情况下, 甚至可以将数据插入到Redis库中。

例如, 可以通过web表单, 将 hash 值取模之后相同的一大批字符串提交到 hash table 中, 这样就可能将时间复杂度为 O(1) 的散列操作, 降级为 O(N) 的最坏情况, 导致 CPU 资源耗尽, 形成拒绝服务攻击(Denial of Service, Dos)。

为了防止这类攻击, Redis 每次启动, 都使用不同的伪随机数种子(pseudo-random seed)来执行 hash 运算。

Redis 的 `SORT` 命令使用了 qsort 算法. 目前为止, 该算法不是随机的, 如果攻击者精心构造一组特定的输入, 在最坏情况下, 可能会造成平方级的时间消耗。

字符串转义和NoSQL注入

Redis 协议中没有字符串转义(escaping)的概念, 所以正常情况下, 不可能通过客户端进行注入. Redis 协议使用的是 prefixed-length 的字符串, 是二进制安全的。

EVAL 和 **EVALSHA** 命令执行的Lua脚本, 也遵循同样的规则, 因此这些命令都是安全的。

但实际情况可能比较复杂, 应用程序应该避免将不受信任来源的字符串, 当做Lua脚本来执行。

代码安全

在一般的 Redis 配置中, 客户端可以执行 command set 中的所有命令, 对实例的访问不太可对 Redis 宿主机的行为造成影响。

Redis内部使用了各种著名的代码安全最佳实践, 以阻止缓冲区溢出(buffer overflow), 格式错误(format bug), 或者其他内存泄露问题(memory corruption). 但是, 控制服务器配置的 **CONFIG** 命令, 有可能修改服务器的工作目录(working dir), 以及 dump 文件的名称. 这就允许客户端将 RDB Redis 文件写入任何路径, 也就造成了 安全问题, 通过启动 Redis 服务的账户权限, 来执行某些操作, 以及某些危险的代码, 甚至有可能造成系统宕机(译者注: 如占满某些目录/磁盘空间等等)。

Redis 不应该使用 root 权限来启动。建议使用非特权的专有账户 `redis`. Redis 作者目前正在尝试, 已决定是否有必要增加新的配置参数, 来阻止 **CONFIG SET/GET dir** 和类似的运行时配置命令. 这能有效阻止客户端将服务器的 dump 文件写到其他目录。

GPG密钥(GPG key)

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
Version: GnuPG v1.4.13 (Darwin)

mQINBFJ7ouABEAC5HwiDmE+tRCsWyTaPLBFEGDHcWOLWzph5HdrRtB//UUlSVt9P
tTWZpDvZQvq/ujnS2i2c54V+9NcgVqsCEpA0uJ/U1sUZ3RVBGfGO/l+BIMBnM+B+
TzK825TxER57ILeT/2ZNSebZ+xHJf2Bgbun45pq3KaXUrRnuS8HWSysC+XyMoXET
nksApwMmFWEPZy62gbeayf1U/4yxP/YbHfwSaldpEILOKmsZaGp8PAtVYMVYHsie
gOUdS/jO0P3silagq39cPQLiTMSsyYouxaagbmtdbwINUX0cjtoeKddd4AK7PIww
7su/lhqHZ58ZJdlApCORhXPaDCVrXp/uxAQfT2HhEGCJDTpctGyKMFXQbLUhSuzf
IilRKJ4jqjcwy+h5lCfDJUvCNYfwyYApsMCs6OWGmHRd7QSFNSs335wAEbVPpO1n
oBJHtOLywZFPF+qAm3LPV4a0OeLyA260c05QZYO59itakjDCBdHwrwv3EU8Z8hPd
6pMNLZ/H1MNK/wWDVeSL8ZzVJabSPTfADXpc1NSwPPWSETS7JYWssdoK+lXMw5vK
q2mSxabL/y91sQ5uscEDzDyJxEPlToApyc5qOUiqQj/thlA6FYBlo1uuuKrpKU1I
e6AA3Gt3fJHXH9TlIcO6DoHvd5fS/o7/RxyFVxqbRqjUoSKQeBzXos3u+QARAQAB
tChTYWx2YXRvcmUgU2FuZmlsaXBwbyA8YW50aXJlekBnbWFpbC5jb20+iQI+BBMB
AgAoBQJSe6LgAhsDBQld/A8ABgsJCAcDAgYVCAIJCgsEFgIDAQIeAQIXgAAKCRAx
gTcoDlyI1riPD/oDDvyIVHtgHvdHqB8/GnF2EsaZgbNuwbiNZ+ilmqnjXzZpu5Su
kGPXAAo+v+rJVLSU2rjCUoL5PaoSlhznw5PL1xpBosN9QzfynWLvJE42T4i0uNU/
a7a1PQCluShnBchm4Xnb3ohNVthFF2MGFRT4OZ5VvK7UcRLYTZoGRlKRGKi9HWea
2xFvyUd9jSuGZG/MMuoslgEPxei09rhDrKxnDNQzQZQpamm/42MITh/1dzEC5ZRx
8hgh1J70/c+zEU7s6kVSGvmYtqbV49/YkqAbhENIeZQ+bCxcTpojEhfk6HoQkXoJ
oK5m21BkMlUEvf1oTX22c0tuOrAX8k0y1M5oismT2e3bqs2OfezNsSfK2gKbeASk
CyYivnbTjmOSPbkvtb27nDqXjb051q6m2A5d59KHfey8BZVuV9j35Ettx4nrS1Ni
S7QrHWRvqceRrIrqXJKopyetzJ6kYDlbP+EVN9NJ2kz/WG6ermltMJQoC0oMhwAG
dfrttG+QJ8PCOlaYiZLD2bjzkDfdfanE74EKYWt+cseenZUf0tsncltRbNdeGTQb
1/GHfwJ+nbA1uKhcHCQ2WrEeGiYpvwKv2/nxBWZ3gwaiAwsz/kI6DQlPZqJoMea9
8gDK2rQigMgbE88vIli4sNhc0yAtm3AbNgAO28NUhzIitB+av/xYxN/W/LkCDQRS
e6LgARAAtdfwe05ZQ0TZYAoeAQXxx2mil4XLzj6ycNjj2JCnFgpYxA8m6nf1gudr
C5V7HDlctp0i9i0wXbf07ubt4Szq4v3ihQCnPQKrZZWfRXxqg0/TOXFfkOdeIoXl
Fl+yC5lUaSTJSg21nxIr8pEq/oPbwpdnWdEGSL9wFanfDUNJExJdzxgyPzD6xubc
OIn2KviV9gbFzQfOIkgkl75V7gn/OA5g2SOLOIPzETLCvQYAGY9ppZrkUz+ji+aT
Tg7HBL6zySt1sCCjyBjFFgNF1RZY4ErtFj5bdBGKCuglyZou4o2ETfA8A5NNpu7x
zkls45UmqRTbmsTD2FU8Id77EaXxDz8nrmjz8f646J0rqn9pGnIg6Lc2PV8j7ACm
/xaTH03taIloOBkTs/Cl01XYeloM0KQwrML43TIm3xSE/AyGF9IGTQo3zmv8SnMO
F+Rv7+55QGlSkfIkXUNCUSm1+dJSBnUhVj/RAjxkekG2di+Jh/y8pkSUxPMDrYEa
OtDoiq2G/roXjVQcbOyOrWA2oB58IVuXO6RzMYi6k6BMpcbmQm0y+TcJqo64tREV
tjogZeIeYDu31eylwijwP67dtbWgiorrFLm2F7+povfXjsDBCQTYhjH4mZgV94ri
hYjP7X2YfLV3tvGyjsMhw3/qLlEyx/f/97gdAaosbpGlVjnhqicAEQEAAYkCJQQY
AQIADwUCUnui4AIbDAUJXfwPAAAKCRAxgTcoDlyI1kAND/sGnXTbMvfHd9AOzv7i
hDX15SSeMDBMWC+8jH/XZASQF/zuHk0jZNTJ01VAdpIxHIVb9dxRrZ3bl56BByyI
8m5DKJiIQWVai+pfjKj6C7p44My3KLodjEeR1oOODXXripGzqJTJNqpW5eCrCxTM
yz1rzO1H1wziJrRNc+ACjVBE3eqcxsZkDZhWN1m8StlX40YgmQmID1CC+kRlV+hg
LUlZLWQIFCGo2UJYoIL/xvUT3Sx4uKD4lpOjyApWzU40mGDaM5+SOsYYrT8rdwvk
nd/efspff64meT9PddX1hi7Cdqbq9woQRu6YhGoCtrHyi/kklGF3EZiw0zWehGAR
2pUeCTD28vsMfJ3ZL1mUGiwlFREUZAcjIlwWDG1RjZDJeZ0NV07KH1N1U8L8aFcu
+CObnlwiavZxOR2yKvwkqmu9c7iXi/R7SVcGQlNao5CWINdzCLHj6/6drPQfGoBS
K/w4JPe7fqmIonMR6O1Gmgkq3Bwl3rz6MWIBN6z+LuUF/b3ODY9rODsJGp21dl2q
xCedf//PAyFnxBNf5NSjyEoPQajKfplfVS3mG8USkS2pafyq6RK9M5wpBR9I1Smm
gon60uMJRIZbxUjQMPLOViGNXbPIilny3FdqbUgMieTBDxrJkE7mtkHfuYw8bERy
vI1sAEeV6ZM/uc4CDI3E2TxEbQ==
```

密钥指纹**(Key fingerprint)**

```
pub   4096R/0E5C88D6 2013-11-07 [expires: 2063-10-26]
      Key fingerprint = E5F3 DA80 35F0 2EC1 47F9  020F 3181 3728 0E5C 88D6
      uid                  Salvatore Sanfilippo <antirez@gmail.com>
      sub   4096R/3B34D15F 2013-11-07 [expires: 2063-10-26]
```

原文链接: https://redis.io/topics/security

翻译日期: 2017年11月30日

翻译人员: 铁锚: http://blog.csdn.net/renfufei